Symposium OPEN IVR
Web Option Guide

Product release: 4.0
Document release: Standard 1.0
Date: July 1998

© 1998 Northern Telecom
All rights reserved

Printed in the United States of America

Information is subject to change without notice. Northern Telecom reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

Symposium, OPEN IVR, and Nortel are registered trademarks of Northern Telecom Limited in the U.S. and Canada. DEC and VT 420 are registered trademarks of Digital Equipment Corporation. HP, LaserJet, and ThinkJet are registered trademarks of Hewlett-Packard Company. X Window System and X are trademarks of the Massachusetts Institute of Technology. Motif is a trademark of Open Software Foundation Inc. UNIX is a registered trademark of Novell, Inc.

Publication history

July 1998 This is the Standard 1.0 issue of the *Symposium OPEN IVR Web Option Guide* for Release 4.0.

Contents

About this guide **ix**
Document organization ix
Supporting Release 4.0 documentation x
Conventions used in this guide xi

1 Web application overview **1-1**
Overview of the Web and IVR applications 1-1
IVR/Web interaction 1-3
A Web server view of Web/IVR interaction 1-5
An OPEN IVR view of Web/IVR interaction 1-6

2 Web system requirements and software installation **2-1**
Overview 2-1
MRS requirements 2-1
AP requirements. 2-1
Web server requirements 2-2
Web browser requirements 2-2
System requirements and configuration 2-2
The license file 2-3
Before you begin 2-3
Installing the License file 2-4
Web software installation 2-5
Media contents. 2-6
Web server installation prerequisites 2-6
Installing the AP software Web option on the Web server 2-7
Post installation adjustments 2-9

3 Web option features and enhancements 3-1

AP software Web option features . . . 3-1
Configuring a virtual Web MRS . . . 3-1
Configuring a range of Web sockets on a Web MRS . . . 3-4
Confirming Web MRS configurations . . . 3-7
Web enhancements to the Application Editor . . . 3-9

4 User issues 4-1

Overview . . . 4-1
Using cells in AP software Web applications . . . 4-1
Parameters supported in the GDAT cell for Web applications: . . . 4-1
Parameters supported in the MENU cell for Web applications: . . . 4-1
Parameters ignored in the PDAT cell by Web applications: . . . 4-2
Cells used in Web and IVR applications . . . 4-2
Ignored cells . . . 4-2
Error causing cells . . . 4-2

5 Developing Web applications for IVR 5-1

Audience . . . 5-1
Web masters . . . 5-1
Application developers . . . 5-1
Developing a mixed media application . . . 5-2
Develop an IVR application . . . 5-2
Develop the Web application . . . 5-2
Integrate the IVR and Web applications . . . 5-2
Stepping through an example . . . 5-3
Developing an IVR application . . . 5-3
Designing a Web page . . . 5-5
Using forms . . . 5-5
Using INPUT fields . . . 5-5
Generalized design procedure for a Web page . . . 5-7
IVR and Web application integration . . . 5-11
Passing data from the Web page to the IVR application . . . 5-12
Developing the call flow specific for the Web page . . . 5-12
Pushing the next template page from the IVR application . . . 5-13
Creating a template page for generating a dynamic Web page . . . 5-14
Generalized design procedure for a Web template . . . 5-16
Suggestions for effective Web designs . . . 5-18
For the Web application . . . 5-18
For the IVR application . . . 5-19

Available application capabilities 5-20
OPEN IVR Common Gateway Interface 5-20
Data dictionaries 5-21
Template Page data fields 5-22
Template Page control constructs 5-27

6 Diagnostics 6-1

Overview 6-1
Testing your Web setup 6-2
Check the results 6-4
OPEN IVR CGI script diagnostics 6-4

Appendix A: Accessing the Service Console Interface A-1

Default account passwords A-1
Using a terminal to access the SCI A-2
Using a modem to access the SCI A-3
Special modem instructions A-3
Accessing the SCI by modem A-3
Using the LAN to access the SCI A-4
Accessing the SCI from an MRS/AP A-5

Glossary Glossary-1

Index Index-1

About this guide

Web integration software is available as a separately purchased option in Symposium OPEN IVR AP software Release 4.0. Web integration functionality enables application developers to integrate with and run AP software applications on the Web.

Document organization

This guide includes six chapters and a comprehensive glossary of terminology used in this guide.

- Chapter 1, "Web application overview," provides an overview of the Web and IVR applications.
- Chapter 2, "Web system requirements and software installation," provides information on Web option system requirements and software installation procedures.
- Chapter 3, "Web option features and enhancements," describes Web option features and enhancements.
- Chapter 4, "User issues," describes user issues you may encounter in using the Web option.
- Chapter 5, "Developing Web applications for IVRprovides guidelines and information on developing Web applications for IVR.
- Chapter 6, "Diagnostics," documents Web option diagnostics and troubleshooting.
- Appendix A, "Accessing the Service Console Interface," provides information about passwords and accessing the SCI.
- "Glossary," provides a list of terms used throughout this book.

Supporting Release 4.0 documentation

Additional information about Symposium OPEN IVR 4.0 is presented in the following Release 4.0 documentation

Document	NT number/ A0 number/
Symposium OPEN IVR 4.0 Getting Started Guide	NTVE8201 A0732100
Symposium OPEN IVR 4.0 Product Guide	NTVE8202 A0732101
Symposium OPEN IVR 4.0 System Administration Guide	NTVE8203 A0732102
Symposium OPEN IVR 4.0 Application Development Guide	NTVE8209 A0732108
Symposium OPEN IVR 4.0 Cell Catalog	NTVE8208 A0732107
Symposium OPEN IVR 4.0 Installation, Upgrade, and Maintenance Guide	NTVE8204 A0732103
Symposium OPEN IVR Planning and Engineering Guide	NTVE8205 A0732104
Symposium OPEN IVR 4.0 Error Messages Guide	NTVE8206 A0732105
Symposium OPEN IVR 4.0 Web Option Guide	NTVE8210 A0732109
Symposium OPEN IVR 4.0 SQL Server User Guide	NTVE8216 A0732115
Symposium OPEN IVR 4.0 Service Console Interface Reference Manual	NTVE8207 A0732106
Symposium OPEN IVR 4.0 VT100 Host Connectivity Configuration and Development Guide	NTVE8221 A0732120
Symposium OPEN IVR 4.0 Text-to-Speech Option Guide	NTVE8213 A0732112

Symposium OPEN IVR 4.0 Speech Recognition Option Guide	NTVE8212 A0732111
Symposium OPEN IVR 4.0 Fax Option Guide	NTVE8211 A0732110
Symposium OPEN IVR 4.0 CAS and ISDN Signaling Option Guide	NTVE8214 A0732113
Symposium OPEN IVR 4.0 Integration Package for Meridian Link User Guide	P0881297 NTVE8215
Symposium OPEN IVR 4.0 Prompt Import Utility Guide	NTVE8218 A0732117
Symposium OPEN IVR 4.0 SNA Host Communications Guide	NTVE8227 A0732774

Conventions used in this guide

Throughout this guide, several typographic conventions have been used to highlight certain types of information:

- Items that appear on the Symposium OPEN IVR screens are identified in quotes, for example:

  the “Timeout” part of the parameters window

- Field names, Prompts, File and Menu names, Directories, Accounts, and Options are shown in a different typeface, for example:

  `/sci>`
  `sci/install menu`
  `/u/nortel/3270/getbalance.act file`

- Symposium OPEN IVR buffer names are shown in all uppercase characters, for example:

  the CURRENT MESSAGE buffer

- Items you must type are shown in bold in a different typeface, for example:

  at the prompt, type **`tn5250`**

- Variables shown in command lines appear in italics, for example:

  the `host_cfg`*`n`* file, where *n* is a variable representing a board number

- Key names you press are shown in angled brackets, for example:
  the <F1> key
- Keyboards usually have keys named <Return> or <Enter> that perform the same function. For convenience, this guide uses the keyname <Enter> to represent both keynames.

Chapter 1: Web application overview

Overview of the Web and IVR applications

The Web Option empowers developers to build applications that are PSTN- and Web-friendly. A single application can interact with PSTN callers and Web users. This elegant design has many advantages, as follows.

1 Applications originally written for PSTN callers can be Web-enabled with minimal effort, with the following additional advantages:

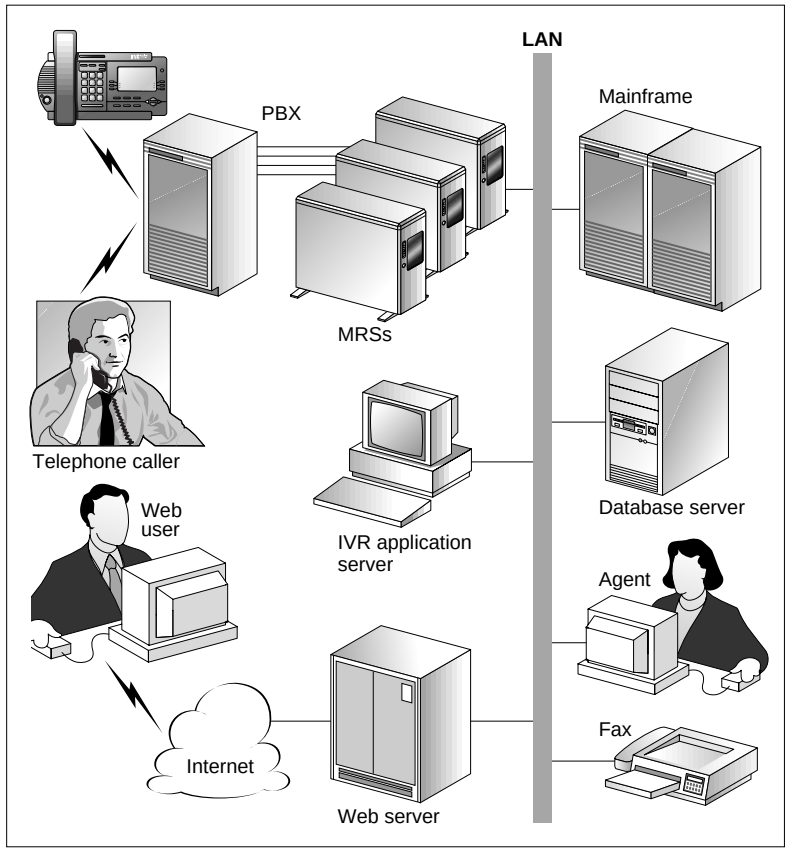
- More addressable customers than PSTN alone
- More satisfied customers as the IVR, Web, and Call Center resources are seamlessly integrated
- Lower capital and operating costs as PSTN hold times are reduced

2 Applications originally written for Web browsers can be PSTN-enabled with modest effort., with the following additional advantages:

- More addressable customers than Web alone
- More satisfied customers, as the IVR, Web and Call Center resources are seamlessly integrated
- Applications can be combined in innovative and powerful ways.

IVR resources (such as Text-to-Speech) and interfaces (such as Host and Database Connectivity) can be made available to Web users. This also provides additional security to the system owner, as the IVR node can effectively act as a firewall between the Internet and their private data network.

Figure 1-1: System overview



To the Web user, the combined Web and IVR environment exhibits the typical behavior of an ordinary Web experience, except that the integration of telephony services provides an enhanced service. The following examples provide information about how a developer can take advantage of combining the two technologies, IVR and Web:

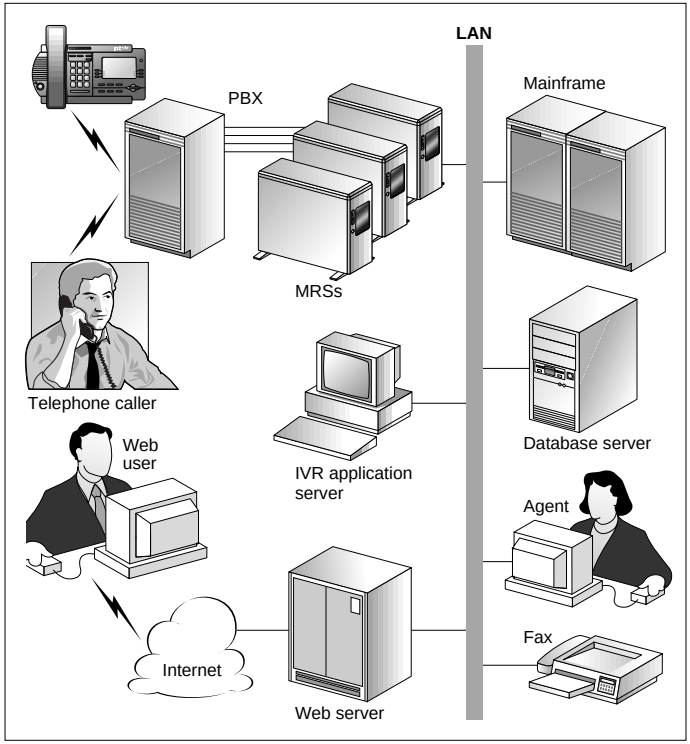
- Receive international callback without an initial call.
- Seek help from a Hotline through the Web, and navigate the Web while waiting for a live response.
- Choose how to communicate with an agent: either over an Internet phone or a telephone.

- View live video or a still image of the agent when contact is made between the agent and the customer.
- Provide minimal training for agents. Interaction between the Internet customer and the telephone customers can appear virtually the same to the agents.
- Minimize programming in developing the two systems since a significant amount of the IVR application can be shared, and, from the customer perspective, is consistent.

IVR/Web interaction

Figure 1-2 illustrates, through a simple IVR/Web application, how the IVR/Web system interacts.

Figure 1-2: IVR/Web interaction



A sample scenario

1 A user navigates the Web and locates a page that offers an item that users can purchase. The fact that this particular page already contains the necessary hooks that link the page to a particular IVR application is transparent to the end user.

2 The user enters data into the appropriate fields in the form and requests a callback. The user submits the order by clicking the mouse on the "Submit" button.

3 The completed form is sent to the Web server.

4 The Web server executes the OPEN IVR Common Gateway Interface (CGI) script.

The contents of the form are passed into the OPEN IVR CGI script. The OPEN IVR CGI script parses the form and places pertinent data into an Input dictionary.

A "call" into the IVR system is established when the OPEN IVR CGI script connects to the application processor through a socket connection.

Once the connection to the application processor is established, the OPEN IVR CGI script passes the Input dictionary to the application processor.

5 The IVR application is run. All non-telephony cells (for example, database cells) operate in the usual manner. The following events occur with telephony cells:

- When input is required from the user through a GDAT or MENU cell, the data passed by the OPEN IVR CGI script from the user's screen input is retrieved from the Input dictionary.
- Data is returned to the user through a PDAT cell. Executing a PDAT cell causes the application processor to write the buffer data contents to the Output dictionary.
- When other telephony operations are encountered, either an error occurs for unsupported cells (for example, the BOUT cell), or the cell is ignored (for example, the PLAY cell). Refer to Chapter 3, Web option features and enhancements" for a list of unsupported cells.

6 When the IVR application is complete, the application processor returns the Output dictionary to the OPEN IVR CGI script. The Output dictionary contains data from any PDAT cells that the call flow executes. The connection between the application processor and the Web server is terminated.

7 The OPEN IVR CGI script generates the dynamic Web page to be displayed back to the user. To generate the dynamic Web page, the OPEN IVR CGI script processes a specified template page and can populate the resulting page with data from either the Input or Output dictionaries. Upon completion, the OPEN IVR CGI script pushes the dynamic Web page to the user's Web browser.

8 In this example, a new page is pushed to the user. It displays a message telling the user that a service representative will be calling.

9 At this point, control is returned to the Web side to await the Web user's next action.

A Web server view of Web/IVR interaction

The Web pages are traversed traditionally, but when IVR functions are required, the OPEN IVR CGI program on the Web server places a "call" to an OPEN IVR application and data is passed between the Web server and the application processor.

The following paragraphs summarize this information.

- Hidden fields are used to store information that is needed in order to integrate with AP software applications. These hidden fields provide information to the Web server such as which OPEN IVR CGI script to execute and which AP software application to invoke. Refer to the section "Using INPUT Fields" for additional information.
- Additional fields are used for each unit of data that is passed to the IVR application (for example, an account number, return phone number, and information about the item in question).
- The OPEN IVR CGI script establishes a socket connection to the application processor. The connection establishes what appears to the user as a telephone call into the AP software system, and the user-supplied data is passed in as DTMF information. This connection is maintained until the call is completed. No action is required by the Web user or the application developer.

An OPEN IVR view of Web/IVR interaction

- The application that is invoked by the OPEN IVR CGI script is a typical OPEN IVR application, which steps through and executes IVR Developer cells.
- When a Web user requests some data through a cell, the IVR Developer cells obtain the data from the application processor. The application processor retrieves the data values from the Input dictionary and automatically places the correct value sequentially into the corresponding cell buffer during execution of the IVR Developer cells.
- When the transactions on the AP software side are completed, you can use an OPEN IVR cell to pass back the name of the next template page. The template page is used by the OPEN IVR CGI script for producing the dynamic Web page which will be presented to the user.
- You can push different template pages throughout the IVR application call flow. This process is useful for providing the user with the appropriate feedback that is dependent on call flow execution. A template page can present to the user any data being returned from the IVR application.

Chapter 2: Web system requirements and software installation

Overview

This chapter includes the following information:

- requirements for
  - Multimedia Resource Server (MRS)
  - Application Processor (AP)
  - Web server
  - Web browser
- system requirements and configuration
- Web software installation

MRS requirements

There are no additional MRS requirements to run the AP software Web option. Refer to the *Symposium OPEN IVR Product Guide* for information about OPEN IVR Release 4.0 requirements.

AP requirements

There are no additional application processor requirements for the Web option. Refer to the *Symposium OPEN IVR Product Guide* for information about OPEN IVR Release 4.0 requirements.

Web server requirements

Requirements for the Web server include the following:

- Unix-based system is recommended; however, enabling technology may be ported to other environments, such as Windows NT
- a separate Web server platform is strongly recommended for runtime use; a second server is optional for development use

  Note: In both cases, a local Web server is not supported for either development or production systems.

- HTTP compliant
- supports HTML Version 2.0 or higher
- supports the Common Gateway Interface (CGI)
- Perl Version 5.0 or higher must be installed
- supports the notation `#!path-name` for the shell where the script is run
- supports a TCP/IP network (LAN) connection between the Web server and the AP
- provides services required to support the Web service

Web browser requirements

The Web browser must support HTML Version 2.0 or higher.

System requirements and configuration

To determine the system requirements for supporting a Web and IVR integrated system, consider the following:

- The load on the Web server is dependent on the number of Web channels available, the support of all active Web applications, and the expected number of Web user hits on your Web site for each application.
- The load on the AP is dependent on the number of Web channels available, the number of MRS channels available, and the consistent traffic flow between the Web server, the AP, and the MRS.

The license file

The license file is unique to your system configuration. Each time your configuration changes in any way, you have to load a new license file to enable its functionality.

Note: The license file is always generated by Nortel—whether for initial installations or for upgrades.

Before you begin

Before installing licenses, make sure of the following:

- the build process has been completed successfully
- the network (between the APs and MRSs) is functioning properly
- remote commands (for example, `rcp`) work properly

Installing the License file

To install the license file, follow these steps.

1 Insert the license diskette into the floppy drive of the main AP.

This diskette is DOS formatted, and contains any of the following files.

File	Contains	For
w1234500.kcd	KDS (version 2.0) enveloped licensed files	AP1
w1234501.kcd		MRS1
w1234502.kcd		MRS2
w1234503.kcd		MRS3
w1234504.kcd		MRS4
w1234588.kcd		MRS/AP
w1234599.kcd		redundant AP (AP2)

The diskette could also contain raw (unenveloped) versions of license files.

File	Contains	For
w1234500.kcd	raw licensed file	AP1
w1234501.kcd		MRS1
w1234502.kcd		MRS2
w1234503.kcd		MRS3
w1234504.kcd		MRS4
w1234588.kcd		MRS/AP
w1234599.kcd		redundant AP (AP2)

Note: Any files named differently from the above will *not* work.

2 At the UNIX prompt, type the following:

install_licenses

3 Monitor the progress of the installation.

If the installation of the licenses to an AP or an MRS was successful, the message `Successfully installed license for...` is displayed.

Note: If there are any problems distributing the files to the different MRSs or APs, the following message is displayed:

```
Unable to copy license file for....
```

This indicates a problem with the network connection. Test your network connections with `ping` or `telnet`, and try again.

If there is a problem unwrapping a KDS enveloped license file, the following message is displayed:

```
Unable to extract license file for....
```

This indicates a likely problem with either the diskette itself or the drive that you are using. Test both of these and then try the installation again.

4 Restart the OPEN IVR system. The license takes effect upon start-up.

Web software installation

The AP software Release 4.0 Web option is preinstalled; however, it must be licensed.

If you are upgrading to AP software Release 4.0 Web option on an existing system, refer to the installation prerequisites and procedure contained in this chapter for installing AP software Release 4.0 Web option support files on the Web server.

The release media consists of a CD-ROM.

This section presents information about the following topics:

- media contents
- installation prerequisites
- installing the AP software Release 4.0 Web option support files on the Web server

Media contents

This is a list of files required for AP software Web support that are installed into the CGI location on the Web server:

- `vtkcgi.cgi` - OPEN IVR CGI script that provides the communication layer between the Web page and AP software.
- `config.pl` - configuration information CGI script used by the script `vtkcgi.cgi`
- `sockdict.pl` - socket dictionary used by the script `vtkcgi.cgi` to pass the dictionary reading to and from the socket
- `webdict.pl` - Web dictionary used by the script `vtkcgi.cgi` to pass the dictionary reading to and from the Web page
- `webutil.pl` - general utilities used by the script `vtkcgi.cgi` and its support files
- `socketerr.html` - socket error page
- `unkappletid.html` - unknown application error page
- `unkreqmeth.html` - unknown request method page
- `unktemplpath.html` - unknown template page specified

Web server installation prerequisites

Ensure that the following prerequisites are met before you install the AP software Release 4.0 Web option.

- The Web server has been Installed and started.
- The Perl software is installed, as required by the OPEN IVR CGI script in the directory `/usr/local/bin`.
- You have read/write access to the Web server (the capability to place the installed scripts on the Web server)
- The installed scripts also need the location of and read/write access to the following:
  - the CGI directory
  - the document directory
- Note the node name of the AP before you begin, as you will be prompted for this name during the installation.

Installing the AP software Web option on the Web server

To install the AP software Release 4.0 Web option, follow these steps.

1. Log in as **root** to the system where you installed your Web server.
2. Change directories (**cd**) to the base directory where you installed the Web server.

   For example:

   ```
   mkdir /usr/local/bin/cgi-bin
   mkdir /usr/local/bin/htdocs
   cd /usr/local/bin
   ```
3. Insert the OPEN IVR Release 4.0 CD-ROM into the appropriate drive of the Web server.
4. At the shell prompt, type the following command:

   ```
   mount /dev/cd/mnt
   cp /mnt/rsp/web/install-web
   ```
5. Press <Enter>.

   The installation file is extracted from the media.
6. Execute the installation process by typing the following command at the prompt (explicitly enter a space before the “dot” command):

   ```
   <space>./install_web
   ```
7. Press <Enter>.

   The following message is displayed at the start of the installation:.

   ```
   Nortel OPEN IVR WEB SERVICES 4.00.00 Installation
   Procedure
   ```
8. The installation procedure prompts you to enter the base directory location of your CGI scripts:

   ```
   Enter CGI binary base directory
   [/usr/local/bin/cgi-bin]:
   ```
   If the default value provided, `[/usr/local/bin/cgi-bin],` is correct, press <Enter>.

   Otherwise, enter the appropriate CGI script base location to override the default location; then press <Enter>.
9. The installation prompts you to enter the base directory location of your Web pages:

   ```
   Enter HTML document base directory
   [/usr/local/bin/htdocs]:
   ```

10 If the default value provided, [`/usr/local/bin/htdocs`], is correct, press <Enter>.

Otherwise, enter the appropriate Web page base location to override the default location; then press <Enter>.

The installation prompts you to enter the host node name of the AP:

```
Enter the AP node name  []:
```

To determine the host node name, run the “hostname” command, or refer to your Web administrator.

11 Enter the node name and press <Enter>.

The installation displays the selections:

```
CGI binary base directory is  ->
/usr/local/etc/httpd/cgi-bin <-
HTML document base directory  ->
/usr/local/etc/httpd/htdocs <-
AP hostname is   -> hostname <-
HTTP Server hostname is  -> serv_hostname <-
Is this correct? [y/n]:
```

12 If you type **n** at the prompt and press <Enter>, the installation displays the following message and the installation is aborted:

```
Please consult your system administrator and run
install again!
```

If you type **y** at the prompt and press <Enter>, then the installation displays the following selections:

```
Please select appropriate action.
1. Extract files from distribution media and
execute configuration script.
2. Execute configuration script without extracting
files.
Please enter your selection or q to quit []:
```

13 Type **2** at the prompt and press <Enter>. The installation assumes that the contents of the media reside in the current directory local to the hard drive.

The following message is displayed:

```
Please wait... Extracting files.
```

Once the extraction is completed, the installation displays the following messages:

```
Uncompressing files...
Updating OPEN IVR WEB SERVICES configuration file
```

The installation is complete. The following message is displayed:

```
Installation Complete
```

Post installation adjustments

After the installation, you need to make site-specific customization to some error messages to ensure that user feedback reaches the correct support personnel. These files all have the .html extension, and are in the /cgi-bin directory.

Table 2-1: Error files needing customization

File	Lines needing to be changed
socketerr.html:	
“The webmaster can be reached at <A HREF="mailto:info@foliage.com">”	
“info@foliage.com</A>. Thank you for your understanding in this”	
unkappletid.html:	
“be reached at <A HREF="mailto:webmaster@your_company.com">”	
“webmaster@your_company.com</A>”	
unkreqmeth.html:	
“be reached at <A HREF="mailto:webmaster@your_company.com">”	
“webmaster@your_company.com</A>”	
unktemplpath.html:	
“be reached at <A HREF="mailto:webmaster@your_company.com">”	
“webmaster@your_company.com</A>”	

Chapter 3: Web option features and enhancements

This chapter presents information about features and enhancements in the AP software Release 4.0 Web option and is organized as follows:

- AP software Web option features
- Web enhancements to the Application Editor

AP software Web option features

The following features and enhancements are discussed in this section:

- configuring a Web MRS
- Web enhancements to the Application Editor

Configuring a virtual Web MRS

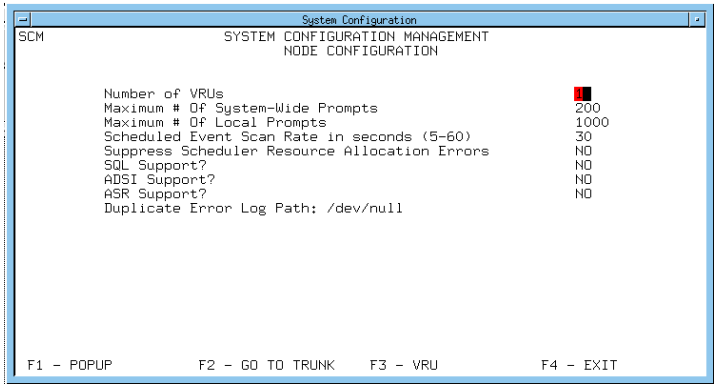
In order for AP software to accept connections from the Web, you need to configure a Web MRS and specify the appropriate number of licensed Web sockets. Refer to the Release 4.0 *Symposium OPEN IVR System Administration Guide* for additional information about using the System Configuration Management (SCM) interface.

> ***Note:*** The Web MRS is a logical server that sits on the AP. It is *not* a physical machine.

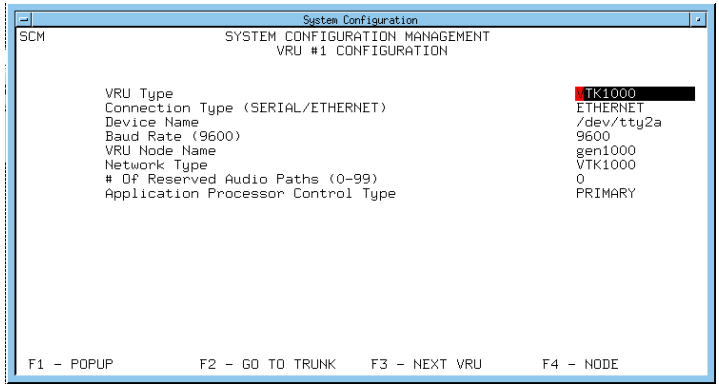
To configure a Web MRS, follow these steps.

1 In the control panel, from the `System Configuration...` pull-down menu, select "`System Configuration Management`" (SCM).

This displays the System Configuration Management Node Configuration screen, as shown below.



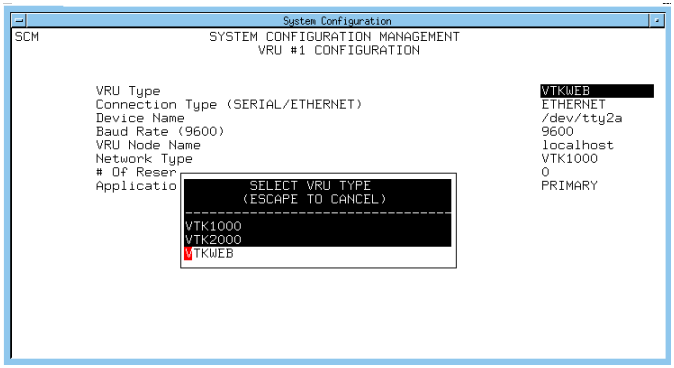
2 In the System Configuration Management Node Configuration screen, press <F3> to access the System Configuration Management MRS Configuration screen, as shown below.



3 Highlight the field MRS Type and press <F1> to display the option popup SELECT MRS TYPE.

4 From the popup SELECT MRS TYPE, select MRS/WEB**.** Press <Enter>.

The MRS configuration screen is displayed, as shown below.



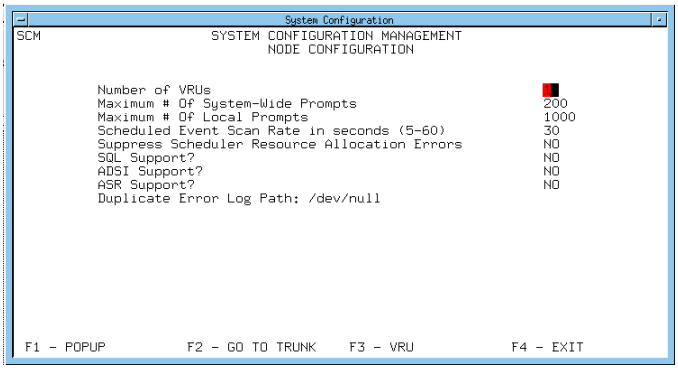
Note: You can ignore other options displayed in the MRS configuration screen when you configure a Web MRS.

Configuring a range of Web sockets on a Web MRS

To configure a range of Web sockets on a Web MRS, follow these steps.

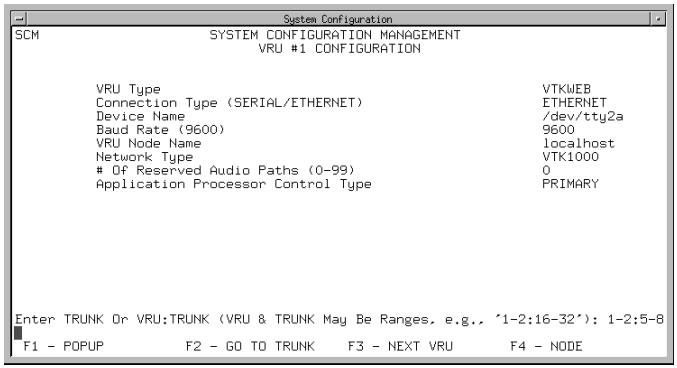
1 In the control panel, from the `System Configuration...` pull-down menu, select "`System Configuration Management`" (SCM).

This displays the System Configuration Management Node Configuration screen, as shown below.



2 From the SCM screen, press <F2> to access the Trunk Configuration screen.

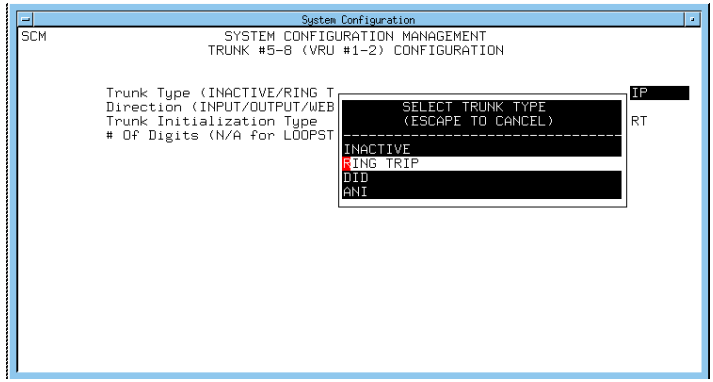
The following prompt is displayed in the Trunk Configuration screen: `Enter TRUNK OR MRS:`.


System Configuration
SCM
SYSTEM CONFIGURATION MANAGEMENT
VRU #1 CONFIGURATION
VRU Type VTKWEB
Connection Type (SERIAL/ETHERNET) ETHERNET
Device Name /dev/tty2a
Baud Rate (9600) 9600
VRU Node Name localhost
Network Type VTK1000
Of Reserved Audio Paths (0-99) 0
Application Processor Control Type PRIMARY
Enter TRUNK Or VRU:TRUNK (VRU & TRUNK May Be Ranges, e.g., '1-2:16-32'): 1-2:5-8
F1 - POPUP
F2 - GO TO TRUNK
F3 - NEXT VRU
F4 - NODE

3 Enter the MRS number for the Web MRS. Then specify a range of trunks to configure as Web sockets; press <Enter>.

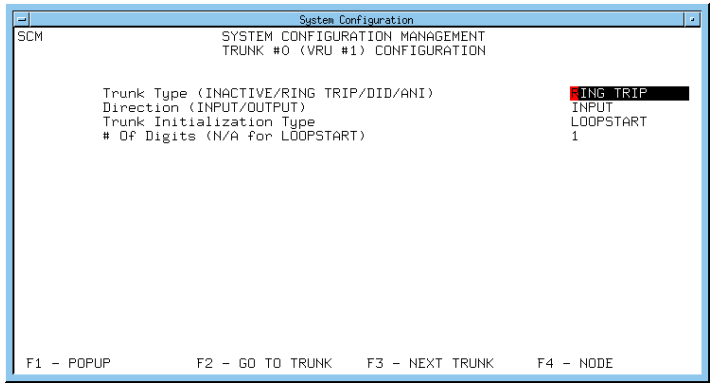
The MRS and trunk range you specify are displayed. Refer to the *Symposium OPEN IVR System Administration Guide* for information about specifying ranges of trunks.

4 In the Trunk Configuration screen, highlight the field `Trunk Type`. Press <F1> to display option popup `Select Trunk Type`, as shown below.

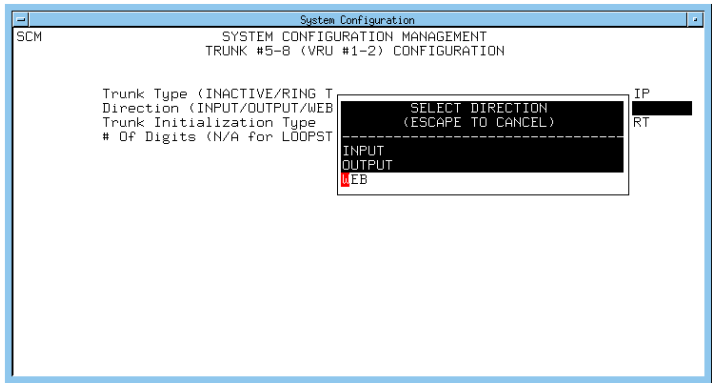


5 From the option popup `Select Trunk Type`, select the option `RING TRIP`. Press <Enter>.

The Trunk Configuration screen is displayed, as shown below.

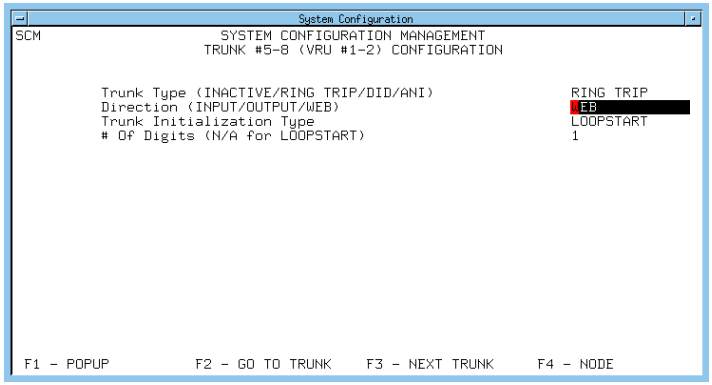


6 In the Trunk Configuration screen, press <F1> to display the `Select Direction` popup. From the popup, select the field `Direction`, as shown below.



7 From the `Select Direction` popup, select `Web`.

The SCM screen below shows the `Direction` field set to `Web`.



8 After you configure the Web MRS with the appropriate number of Web sockets, save the configuration and press <F4> to exit from the SCM interface.

Note: The SCM verifies the validity of the field values when you select the MRS type, MRSWEB. For instance, if you select MRSWEB and then you configure a number of trunks as sockets, the SCM saves the configured trunks as Web sockets. You see the changes to the configuration the next time you invoke the SCM interface.

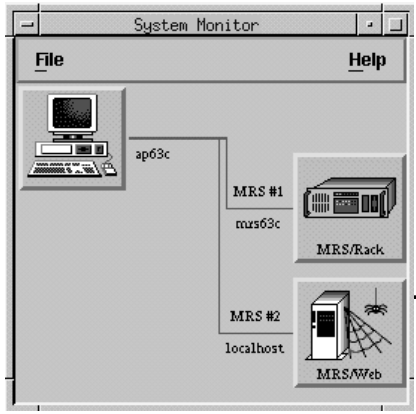
Confirming Web MRS configurations

To confirm the Web MRS configuration and the number of Web sockets within the AP software, follow these steps:

1 Start the AP software.

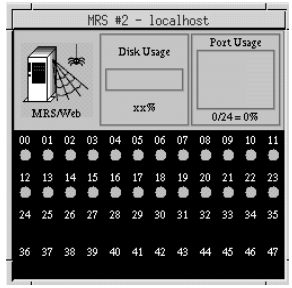
2 Start the System Monitor by selecting `System Monitor...` from the pull-down menu in the control panel.

The System Monitor Administration window is displayed. It shows a Web and spider in the MRS/WEB icon, as shown below.



3 With MB1, click the MRS/WEB button.

A pop-up window is displayed which indicates the trunks you configured as Web sockets for the MRS type, MRS/WEB, as shown below.



This MRS/WEB popup shows an example of a Web MRS dialog box. When the sockets are configured properly and the licenses for the sockets are obtained, the sockets are displayed as gray circles; the gray circles indicate the availability of the sockets. However, at this stage, no activity is taking place on these sockets.

Compare the MRS/Web popup shown in step 3 with the MRS/Rack popup, as shown in Figure 3-1.

Figure 3-1: MRS/Rack popup

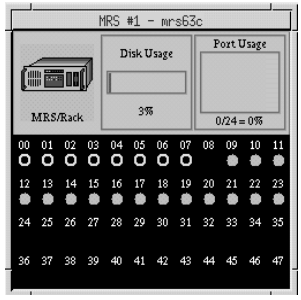
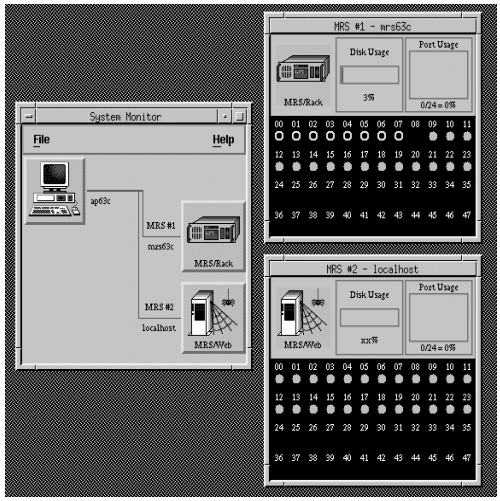


Figure 3-2 illustrates a Web MRS and an MRS/Rack configuration in the MRS dialog window.

Figure 3-2: Web MRS and MRS/Rack dialog box



Web enhancements to the Application Editor

The system buffer ISWEB enables application developers to create an AP software application and identify through this buffer whether AP software is receiving a call from the Web application or from a telephony MRS.

The application branches according to the value set in the system buffer. The default value for the buffer ISWEB is `0` (zero), which indicates that AP software is receiving a call from a telephony MRS. If the ISWEB buffer contains a value of 1, then AP software is receiving a call from a Web MRS.

> ***Note:*** Specify the value in the buffer ISWEB as read-only in your AP software application.

Chapter 4: User issues

Overview

This chapter presents information about the following topics:

- using cells in AP software Web applications
- known conditions and workarounds

Using cells in AP software Web applications

The source of input for the MENU and GDAT cells used in Web applications is the data dictionary, which is received from the Web page. Otherwise, these cells perform identical functions in both IVR and Web applications.

Parameters supported in the GDAT cell for Web applications:

- Call Audit Enabled
- Call Audit Information
- Input Digit

The other parameters in the GDAT cell are ignored.

Note: Prompts are not played in the GDAT or MENU cells for Web applications.

Parameters supported in the MENU cell for Web applications:

- Call Audit Enabled
- Call Audit Information
- Input Digit

The other parameters in the MENU cell are ignored.

Parameters ignored in the PDAT cell by Web applications:

- Abort Play On Input/Allow Overdialing
- Number of Prompts

The **HANG cell** is typically the last cell in each AP software Web application.

Cells used in Web and IVR applications

Table 4-1 lists the cells that are used for Web and IVR applications. Most of these cells do not rely exclusively on telephony resources, and can be used in both IVR and Web applications.

Ignored cells

Some cells are ignored during the execution of an application during a Web call. These cells are marked with the superscript "I" (^{I}) in Table 4-1.

Error causing cells

Other cells generate errors when used in AP software Web applications, and their functionality is not available to any AP software Web applications. These cells are marked with the superscript "E" (^{E}) in Table 4-1. Avoid using these cells for any Web call-flow handling.

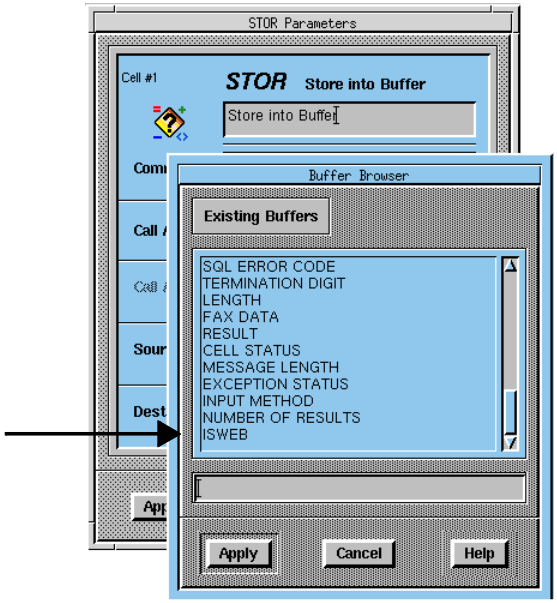
> ***Note:*** You can include these cells in your AP software Web application. Only allow execution of these cells within the application if the buffer ISWEB is not set to `1` (one). Use a COMP cell to check whether the buffer ISWEB is equal to `1` (one). If the buffer is not equal to `1` (one), then the application is executed on a telephony channel and you can use these cells. Figure 4-1 shows the system buffer ISWEB.

.

Table 4-1: Cells used for Web and IVR applications

AMSG [E]	ANSW [I]	BOUT [E]	CASE
CDAT	CEND [E]	CHEK	CMSG [E]
COMA	COMI	COMO	COMP
CONC	CONP [E]	COUT [E]	DCNP [E]
DDEL	DELV	DIAL [E]	DINS
DMSG [E]	DOUT [E]	EVENT	EXEC
GAIN [E]	GMSG [E]	GSUB	LANG
LDLV	MATH	MSGS [E]	PLAY [I]
PMSG [E]	QCNT	QDEL	QINS
QSEL	QUPD	RECP E	RFAX [E]
RMSG [E]	SAY [I]	SFAX [E]	STOR
STPT [I]	SUBS	TIME	TONE [I]
UDLV	UMSG [E]	UPDT	UPWD [E]
USER	VINP [E]	XFER [E]	

Figure 4-1: ISWEB system buffer



Chapter 5: Developing Web applications for IVR

Audience

This chapter is provided for two primary audiences: Web masters and application developers.

Web masters

Nortel assumes that a Web master is familiar with the following:

- purchasing, installing, and using a Web server
- purchasing, installing, and using a Web browser
- the Common Gateway Interface (CGI)
- installing CGI executables onto an existing Web server

Application developers

Nortel assumes that an application developer is familiar with the following:

- can develop AP software applications for Interactive Voice Response (IVR) systems
- understands AP software application management activities and terminology
- can develop HTML pages using common authoring tools
- understands Web terminology and how to relate a Web page to an IVR application using the appropriate HTML tags and IVR Developer cells
- knows how to install and use a Web browser

Developing a mixed media application

Developing an OPEN IVR Web application requires three steps:

1. Develop an IVR application or reuse sections of call flow from an existing application.
2. Develop a Web application using the Web authoring tool of your choice that supports HTML Version 2.0 or greater.
3. Integrate the IVR and Web applications.

Develop an IVR application

Whether you have already developed the IVR application or are in the process of developing a telephony interface, the process is the same—develop an intuitive and easy-to-use IVR application.

Develop the Web application

Develop the Web page in the usual way. Any of the professional Web development tools are adequate for developing a Web page that can be integrated with the OPEN IVR system: such as Microsoft FrontPage 97, HotDog Web Editor Pro, Netscape Navigator Gold, or HoTMetaL.

While you develop a Web application for use with an IVR system, keep in mind that an IVR system provides services such as host computer data retrieval and updating, database interfacing, and access to outgoing telephone lines, that are frequently used in telephony applications. By taking advantage of these services, you can simplify Web development by relying on the tasks that the IVR system already performs.

Integrate the IVR and Web applications

Once you develop both the Web application and the IVR application, you need to link the two systems together. By adding fields to the HTML page, data is passed between the Web page and the IVR system. Modify the IVR application to isolate the Web-related aspects of the IVR system, such as sending Web page references and data values back to the Web server.

Stepping through an example

The goal of the information in this section is to provide you with a step-by-step example that can be used as a guideline for integrating Web and IVR applications. Though the example is simple, the intent is to point out key pieces of information needed for successful application integration. The following is a description of the problem that the example will solve.

The IVR application example obtains a four-digit account number from the caller. The call can be initiated from the telephone or from the Web. The IVR application looks up the account number in an information database. If the IVR application finds the account number, the following events occur:

- The name and phone number are retrieved from the record in the information database.
- Another IVR application is scheduled to perform a callback to the user using the retrieved phone number.
- The IVR application informs the user that a callback is scheduled at the specified phone number.
- The call is completed.

Developing an IVR application

Develop an application that provides the following functionality.

1 Answers the call (ANSW cell).

2 Obtains the account number (GDAT cell).

3 Verifies the account number in an information database and retrieves the phone number (CHEK cell).

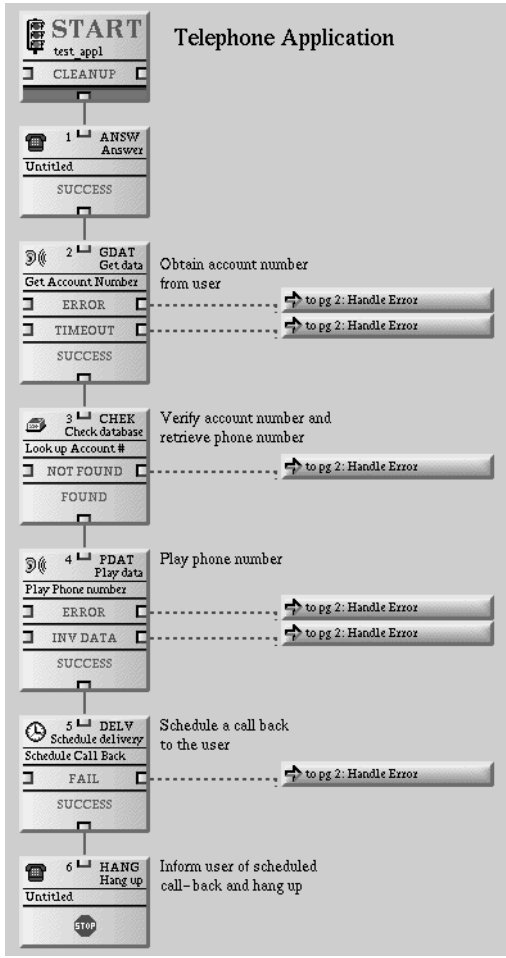
4 Plays the phone number back to the caller (PDAT cell).

5 Schedules a callback IVR application (DELV cell).

6 Informs the user that a callback is scheduled and hangs up the phone (HANG cell).

Figure 5-1 illustrates a simple application for a caller using the phone.

Figure 5-1: Telephone application



Designing a Web page

This section presents some of the important details needed for designing a Web page. Design the initial Web page with the intention of integrating with an IVR application. The Web page is static because the page resides on the Web server and the contents will not change. The Web page provides the hooks to initiate a call into the application server.

You need to understand the following key points in order to successfully initiate a call into the application server and pass data from the Web page to the IVR application:

- using forms
- using INPUT fields
- submitting the form

Using forms

In the Web page, all data to be passed to the OPEN IVR CGI script is contained in a form. In HTML, a form is denoted by the tags `<FORM>` and `</FORM>`. Set additional form parameters as follows:

- METHOD is set to POST.
- ACTION points to the OPEN IVR CGI script, `vtkcgi`. The `vtkcgi` script is installed on your Web server during the AP software Web option installation.

The form and its parameters are supported by all Web authoring tools that use HTML Version 2.0 or greater.

Using INPUT fields

INPUT is the field used to identify the data within the form. The data can be provided either by the user or statically set within the Web page. All data is passed to the OPEN IVR CGI script. The OPEN IVR CGI script parses the form and generates an Input dictionary. The Input dictionary contains all data intended for the IVR application, especially these two types of data:

- data required by the OPEN IVR CGI script
- IVR application data

Data required by the OPEN IVR CGI script.

The OPEN IVR CGI script expects the following field data to be provided by the form:

- `applicationid`

  Identifies the associated IVR application that executes as a result of servicing the Web call. The name of the application is the same name that is loaded by the Application Manager, but reference to the file extension (`.vpf`) is omitted.

- `dfltnextpage`

  Identifies the name of the template page used as a default by the OPEN IVR CGI script for generating the dynamic Web page to be displayed to the user. The default is used only when the IVR application does not push the next template page from within the call flow.

IVR application data

Data destined for the IVR application is identified by the following Input Data syntax:

`xnnn_<label>`

where

- the name must begin with a lowercase `x`
- the field `nnn` is a numerical ordering of the input data. The IVR application expects 000 to be the first input data, 001 the second input data, and so on.
- the underscore (_) is required
- the `<label>` field can be any additional label you want to provide. This is not a required field but will improve the readability of your Web page and, therefore, is highly recommended.

For the example, the Web page provides a text box to the user for collecting the four-digit account number. The text box is contained in a form and is given the name `x000_acct`. Since the name given to the text box follows the Input Data format, the data is tagged for the IVR application.

Any additional data for the IVR application is sequentially tagged within the Web page using the Input Data naming convention. The next piece of data in the example is assigned the name `x001_<label>`, and so on.

Submitting the form

The Web page provides a mechanism for submitting the form to the specified ACTION. The ACTION initiates a call from the Web through the OPEN IVR CGI script, and into the IVR application itself.

In the example, a pushbutton is created and its type is set to submit. When the user enters the account number and clicks on the button with the mouse, the contents of the form are sent to the entity specified by ACTION. In this example, ACTION points to the OPEN IVR CGI script.

Generalized design procedure for a Web page

The following generalized steps outline the process of setting up a Web page that will be used to interact with the IVR application. In this example, a page accepts data into an input field, and sends it from the page to the IVR application.

The steps will present the major pieces which need to be in place for a successful transfer of data from the Web page to the application.

> ***Note:*** The details of individual steps may vary depending on the HTML editor you use to create the page.

1 Invoke your HTML editor.

2 Create a new blank page.

3 Create a form with a text box in it.
This will be used to capture the data.

4 Configure the form settings.
Set the `Action` field to the program on the Web server where the form contents will be sent (for example, specify the OPEN IVR CGI script), and the `Method` field contains the method for sending the information to the server (for example, POST). Click OK to accept the changes.

5 Set the Hidden Fields section of the Form Properties.
Add the hidden fields `applicationid` and `dfltnextpage` to the form. Note that the field `dfltnextpage` points to a template page relative to the base path of the HTML documents directory. The base path of the HTML document directory is set up during the installation of the AP software Web option.

6 Name the first input field.
Change the `Name` field to `x000_first`. Retain any default settings for the rest of the fields.

7 Name the second input field.
Change the `Name` field to `x000_second`. Retain any default settings for the rest of the fields.

8 Tell the system when to submit the input data.
Create a `submit` button. This tells the system to send the gathered information to the address specified in the form properties (see step 4).

9 Optionally embellish the page.
At this point, you may enhance your page with graphics, added text, or a page title.

10 Save the Web page.
Install the Web page on the Web server in the appropriate HTML document directory.

The sample Web page is completed and contains the necessary IVR tags.

Sample input

This example prepares two input text fields, along with a button to submit and reset their data.

```
<HTML>
<HEAD>
<TITLE>NTTEST INPUT</TITLE>
<!-- Background white, links blue (unvisited), navy (visited),
   red (active) -->
</HEAD>
<BODY TEXT="#000000" BGCOLOR="#00A0A0" LINK="#0000FF"
   VLINK="#000080" ALINK="#FF0000">
<CENTER>
<H1>
NTTEST INPUT WEB Page</H1></CENTER>
<HR WIDTH="50%" SIZE="8">
<HR>
This sample form takes two integers as input.  When the submit
   button is pressed, the integers are sent to the NTTEST_WEB
   application where they are added together. The sum is
   displayed in the NTTEST RESULT WEB page.<P>
<P>
<TABLE BORDER=4 CELLSPACING=2 CELLPADDING=4 COLS=1 WIDTH="100%"
   BGCOLOR="#666666" BORDERCOLORLIGHT="#0000FF"
   BORDERCOLORDARK="#0000A0" >
<TR>
<TD>
<H2>
<FORM ACTION="http://localhost/cgi-bin/vtkcgi.cgi"
   METHOD="POST">
<INPUT TYPE="hidden" NAME="applicationid" VALUE="NTTEST_WEB">
<INPUT TYPE="hidden" NAME="dfltnextpage" VALUE="default.html">
<CENTER>Enter the first number
<INPUT TYPE="text" NAME="x000_first" SIZE="5"
   MAXLENGTH="4"VALUE="0" ></CENTER><P>
<CENTER>Enter the second number
<INPUT TYPE="text" NAME="x001_second" SIZE="5" MAXLENGTH="4"
   VALUE="0" ></CENTER><p>
<CENTER><INPUT type="submit" value="Submit" name="B1">
<INPUT type="Reset" value="Set to default" name="B2"></CENTER>
</FORM>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Determining the target display

The NTTEST_WEB Web pages are currently set to display on the local host, specifically the AP of the IVR. To display the Web page on another Web server (that is, on a machine other than the local host), do the following.

1. Copy the file /u/nortel/www/htdocs/index.html to /u/nortelwww/htdocs/index.bak
2. Begin an editing session of file */u/nortel/www/htdocs/index.html*
3. Modify the line

   ```
   <FORM ACTION="http://localhost/cgi-bin/vtkcgi.cgi"
   METHOD="POST">
   ```

   by changing the reference "localhost" to the IP address of the machine on which the Web page is to be displayed.
4. Save the changes.
5. Exit the editing session.

You can reset the Web page to display on the local host using one of the following ways:

- Repeat the above procedure, but this time replace the IP address with "localhost".
- Rename the *index.bak* file as *index.html*.

Sample output

Figure 5-2 shoes how the sample input code on page 5-9 is represented on a Web page.

Figure 5-2: NTTEST input Web page

NTTEST INPUT WEB Page
This sample form takes two integers as input. When the submit button is pressed, the integers are sent to the NTTEST_WEB application where they are added together. The sum is displayed in the NTTEST RESULT WEB page.
Enter the first number
3
Enter the second number
11
Submit
Set to default

IVR and Web application integration

To ensure the successful integration of the IVR application with the Web page, you will need to develop mechanisms for the following:

- passing data from the Web page to the IVR application
- developing the call flow specific for the Web page
- pushing the next template page from the IVR application
- creating a template page for generating a dynamic Web page

Passing data from the Web page to the IVR application

For each INPUT tag named `xnnn_<label>`, the OPEN IVR CGI script creates an entry in an Input dictionary as part of parsing the submitted form. The application developer ensures that for every data unit found in the Input dictionary, the IVR application provides a corresponding GDAT or MENU cell. If the call is invoked from the Web, the call-flow execution places the contents of the Input dictionary into the appropriate buffer of the GDAT or MENU cell being executed.

Note the following:

- The total number of GDAT or MENU cells must match the total number of INPUT `xnnn_<label>` tags found in the Web page.
- Ordering is important. For example, `x000_acct` is specified as an INPUT in the Web page; `x000_acct` is mapped to the first GDAT cell in the IVR application. The next GDAT cell is passed to get `x001_xyz`.

Developing the call flow specific for the Web page

OPEN IVR software provides a mechanism for the application developer to distinguish a Web call from a telephone call. When a new call is received by the application server, AP software sets a system buffer, ISWEB, to 1, if the call originated from the Web. Otherwise, ISWEB remains at a default setting of 0. Use of ISWEB helps distinguish between the two call types in the IVR application. The call from the Web takes a different path to the application server than a call from the telephone. The application developer can access the ISWEB system buffer from within the applications.

Refer to the Release 4.0 *Symposium OPEN IVR Cell Catalog* for additional information about the ISWEB system buffer.

Pushing the next template page from the IVR application

You can design the IVR application to push a template page. The implementation is described in the following two steps.

1 Store the name of the next template page into a buffer using the following format:

"page:*path/page_name.ext*"

where

"page:" is literal and required.

path/

Path is relative to the HTML document base path location set during the installation of the AP software Web option. For example, if the base path of the HTML document is located in the directory /usr/local/httpd/htdocs, but the template page resides in the directory /usr/local/httpd/htdocs/Demo/foo/, then *path* is set to Demo/foo.

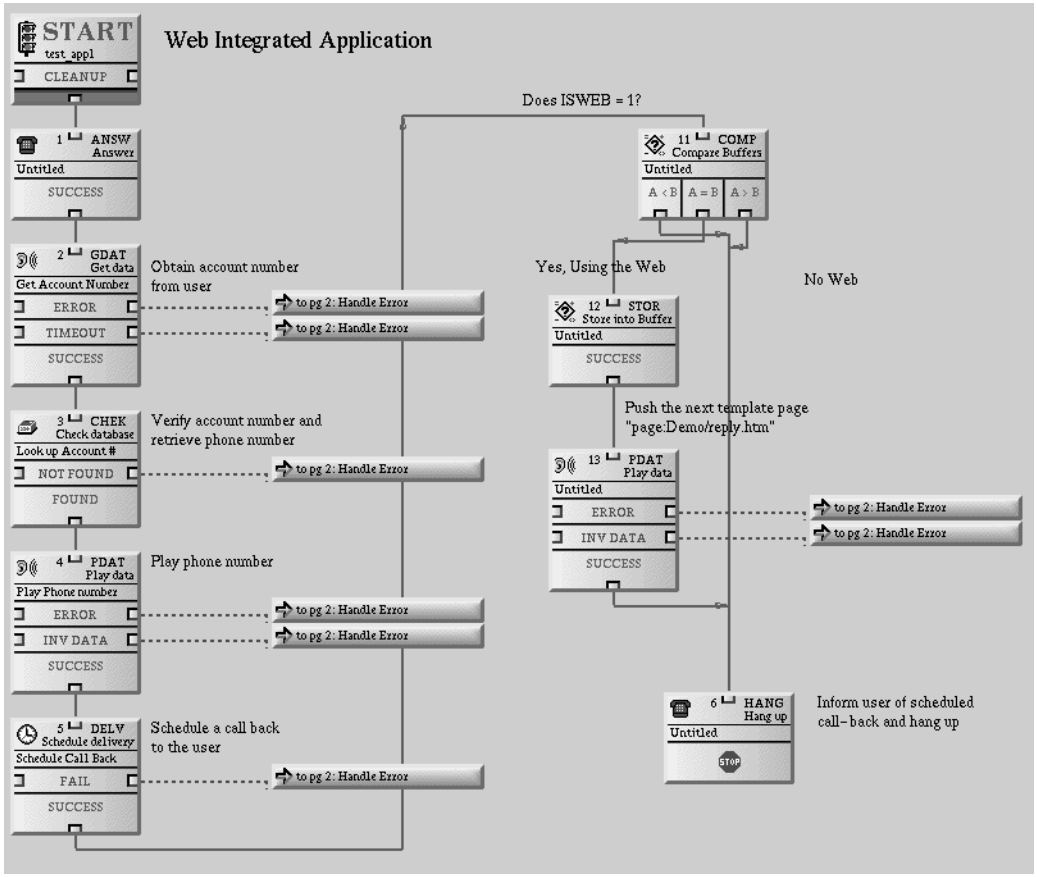
page_name.ext

Name of the file containing the next template page. For example, reply.html.

2 Send the contents of the buffer that contains the name of the next template page to the OPEN IVR CGI script by using a PDAT cell.

Figure 5-3 displays the original IVR application with some minor modifications. Call flow is added to check the ISWEB system buffer and to push the next template page if the call is from a Web application.

Figure 5-3: Web integrated application



Creating a template page for generating a dynamic Web page

The Web page pushed by the IVR application is called a template page. The OPEN IVR CGI script uses the template page for generating a dynamic Web page. The dynamic Web page is displayed back to the user. The application developer provides a template page that outlines the contents of the dynamic Web page. The contents may be graphics, text, data from either the input or output dictionaries, and so on.

The IVR application passes data back to the OPEN IVR CGI script where data can be the name of the template page, data retrieved from a database, or any other pertinent information collected throughout the call flow. Data is passed back through the PDAT cell. For each PDAT cell executed during the call flow, the contents of the buffer specified in the PDAT cell are stored into an Output dictionary. Note that the data is placed into the Output dictionary in the same order that the PDAT cells are executed. Upon completion of the Web call, the contents of the Output dictionary are returned to the OPEN IVR CGI script.

The OPEN IVR CGI script creates the dynamic Web page based on the template page. As the template page is parsed, the OPEN IVR CGI script determines if data needs to be retrieved from either the current Input dictionary (created from the initial static Web page) or the Output dictionary. Once the template page is populated, the resulting dynamic Web page is generated and pushed to the user's browser.

To access the Input dictionary data from within the template page, the developer uses the same Input Data syntax explained in the Using INPUT Fields section. For example, to obtain the account number specified by the user, the template page retrieves this information by using the original name tag and the following syntax:

```
&$x000_acct,
```

By adding `&$` to the beginning of the name tag and adding a comma to the end of the name tag, the OPEN IVR CGI script identifies this as a special macro, Input Data Macro. While parsing the template page, the OPEN IVR CGI script replaces all occurrences of `&$x000_acct`, with `1234` (in this example). Since the name begins with an x, the OPEN IVR CGI script looks for the data in the current Input dictionary.

To populate the template page with data from the Output dictionary, an Output Data Macro is used. The macro uses the following syntax:

```
&$ynnn_<label>,
```

where

- the name begins with a lowercase y
- `nnn` is a numerical ordering of the output data. The IVR application maps the first PDAT cell executed with `000` in the Output dictionary, and so on.
- the underscore (_) is required

- <label> can be any additional label you want to provide. This field is not required but improves the readability of your template page and therefore is highly recommended.

By adding &$ to the beginning of the name tag and adding a comma to the end of the name tag, the OPEN IVR CGI script replaces all occurrences of &$y000_<label>, with the appropriate corresponding data stored in the Output dictionary. In the example, &$y000_phone, is replaced with 5551239876.

Generalized design procedure for a Web template

The following generalized steps outline the process of setting up a template page that will be sent back to the Web user by the CGI script. These steps need to be in place for a successful transfer of data from the application back to the user.

Completing the previous example, the procedure creates a template to accept data into an input field, and outputs data into an output field. The data is then sent it from the page to the IVR application.

Note: The details of individual steps may vary depending on the HTML editor you use to create the page.

1 Invoke your HTML editor.

2 Create a new blank page.

3 Position and compose any lead-in text for your page.
This could be page title or other applicable information.

4 Type the result statement/message.
Any input fields with information taken from the input dictionary should be represented in the form &$x000_<label>,. Present corresponding output fields as &$y000_<label>,.

Note: If you literally type an ampersand into your page, your HTML editor may store an HTML representation instead, "&". The OPEN IVR CGI script does not recognize "&" and data is not retrieved. Use the HTML Markup feature of your editor to force it to accept the ampersand as &.

Since the HTML Markup represents only the ampersand portion of the Output Data macro, you need to provide the remainder of the information to complete the Output Data macro. Place the cursor after the HTML Markup and type **$y000_phone,**.

5 Optionally embellish the page.
At this point, you may enhance your page with graphics, added text, or a page title.

Sample input

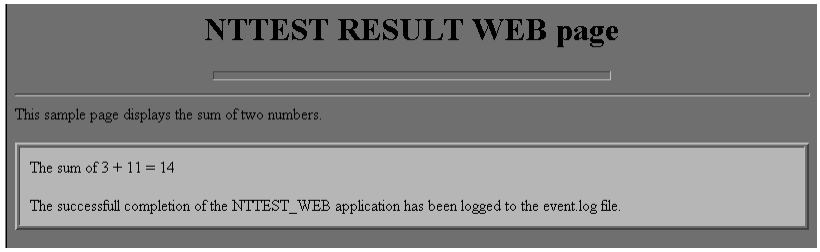
This example presents the results of two input text fields being added together.

```
<HTML>
<HEAD>
<TITLE>NTTEST RESULT</TITLE>
<!-- Background white, links blue (unvisited), navy (visited),
   red (active) -->
</HEAD>
<BODY TEXT="#000000" BGCOLOR="#00A0A0" LINK="#0000FF"
   VLINK="#000080" ALINK="#FF0000">
<CENTER>
<H1>NTTEST RESULT WEB page</H></H1></CENTER>
<HR WIDTH="50%" SIZE="8">
<HR>
This sample page displays the sum of two numbers.
<p>
<P>
<TABLE BORDER=2 CELLSPACING=2 CELLPADDING=10 COLS=1 WIDTH="100%"
   LENGTH="50" BGCOLOR="#ABCABC" BORDERCOLORLIGHT="#0000FF"
   BORDERCOLORDARK="#0000A0" >
<TD>
<p>
<p>
<p>
The sum of  &$x000_first, +  &$x001_second,  =  &$y000_sum,
<p>
<p>
<p>
<p>
<p>
The successfull completion of the NTTEST_WEB application has been
   logged to the event.log file.
<p>
<p>
</TD>
</TABLE>
</BODY>
</HTML>
```

Sample output

Figure 5-4 shoes how the sample input code on page 5-17 is represented on a Web page.

Figure 5-4: NTTEST result Web page


NTTEST RESULT WEB page
This sample page displays the sum of two numbers.
The sum of 3 + 11 = 14
The successfull completion of the NTTEST_WEB application has been logged to the event.log file.

Suggestions for effective Web designs

For the Web application

- Create a page layout that is intuitive for the Web user.
- Avoid retrieving database or host communications data. The IVR system can include these processes.
- As you develop your Web application, consider how the Web user can take advantage of the IVR system.

Example

1 If a callback is expected from an operator, show a picture of the operator that will be calling and tell users the expected delay.

2 Provide a fax mechanism using the same fax resources that you already use in the IVR system. This strategy saves telephone lines, reduces the amount of fax hardware, and simplifies administration tasks.

3 In the case of callbacks, ask users if they want:

a a personal callback on the phone number they chose which may require them to log off their Internet connection

b an Internet phone which allows them to maintain their Internet connection until the operator is ready

Web calls can carry more complex data

Web calls can pass data that is more complex than the data which most applications have become accustomed to handling.

For example, Web calls ignore the termination digit. A user could therefore successfully input `1#2#3#` into a Web page, and the system will pass this full string into the buffer of the application (even if the GDAT cell has been set with # as the termination digit). A customary call flow would not get input in this format. Web calls can also pass double characters from a double character system, and international characters could be placed into the buffer.

As a result, Web applications must be created to handle odd input, and existing applications must be updated to handle this odd input.

For the IVR application

- In general, practice good application programming style and plan your application before developing it.
- Reuse information, as appropriate. Isolate host computer and database interaction to subroutines so they are easily used between telephony IVR applications, and IVR and Web integrated applications.
- Take advantage of the MENU cell. For Web applications, the MENU cell can be used for routing the call to different segments of the call flow. The call-flow segments should be efficient to ensure quick turnaround of channels under load.
- Decide when it is appropriate for IVR and Web applications to use identical call flows.
- Isolate Web call flow in a subroutine to allow for modifications without bringing down the entire IVR application.
- Control the flow of the Web application from the IVR system and pass back specific Web pages to display as the user navigates through the Web pages.

Available application capabilities

The following application capabilities are accessible to Web and OPEN IVR application developers:

- Any IVR outbound agent application can be scheduled for execution.
- Any callback or conferencing application can be initiated.
- Any outgoing fax-back application can be run.
- Host computer access can be performed as in existing IVR applications.
- Databases can be accessed using existing methods.
- Web pages can be pushed to the user.
- Internet Protocol (IP) Web phone calls can be initiated.

Note that a channel configured for the Web is unable to support any bridging or direct outbound call functionality since the Web channel is not a connection to a physical telephone line.

OPEN IVR Common Gateway Interface

The Common Gateway Interface (CGI) is a mechanism that resides on the Web server to allow a Web client to interact with the Web server. The primary uses for a CGI program follow:

- easy execution by the Web client
- ability to process input from an HTML form
- converts information into a format that can be sent back to the Web client
- allows for dynamic Web page generation

The role of the OPEN IVR CGI program

The OPEN IVR CGI provides the communication needed by AP software to allow data to flow between a Web client (the Web page) and an IVR application. The OPEN IVR CGI is accessed through the Perl script, `vtkcgi.cgi`.

The `vtkcgi` script performs the following tasks:

- accepts data contained in a form submitted by a Web page
- processes data contained in a form and passes the data to the application server
- receives data from the application server
- dynamically generates a Web page based on a specified template page. Populates the dynamic Web page using available data dictionaries

Data dictionaries

Data dictionaries are used as a means for communicating data between the Web pages and the IVR applications. The two types of dictionaries are Input and Output.

Input dictionary

The Input dictionary is a collection of NAME/VALUE pairs resulting from the `vtkcgi` script processing of a submitted form. The Input dictionary is the mechanism used for passing data from the Web page into the corresponding IVR application. Data is designated for the Input Dictionary by using the Input Data naming convention, `xnnn_<label>`, as described in the Using INPUT Fields section. During processing of the template page, the `vtkcgi` script can populate the dynamic Web page with data stored in the Input Dictionary.

Each static Web page that submits a form causes the `vtkcgi` to generate a new Input dictionary.

Output dictionary

The Output dictionary is a collection of NAME/VALUE pairs resulting from the IVR application execution of PDAT cells. The Output dictionary is the mechanism used by the application server for returning data back to the `vtkcgi` script. Processing the template page can result in the `vtkcgi` script populating the dynamic Web page with data stored in the Output dictionary.

Each IVR application execution causes a new Output dictionary to be generated and returned back to the `vtkcgi` script.

Template Page data fields

When the `vtkcgi` script processes a template page to produce the dynamic Web page, special macro fields are recognized and executed by the `vtkcgi` script. These indicate to the `vtkcgi` script information such as when to populate with data from either the Input or the Output dictionaries. The following information describes the different macros available to the development of the template page.

Simple Data macro

The syntax used for the Simple Data macro is

`&$`*`name`*`,`

where *name* is one or more alphanumeric characters.

To use a Simple Data macro, the name must be defined as a NAME attribute and assigned a VALUE attribute in the corresponding static Web page. When processing the template page, the `vtkcgi` script executes any occurrence of a Simple Data macro.

If the NAME/VALUE pair exists in the Input dictionary, the macro is replaced by the VALUE. Otherwise, the macro reference is removed and a value is not referenced in the resulting dynamic Web page.

For example, in a static Web page, the following information is contained within a form:

```
<INPUT TYPE=hidden NAME="cgibase"
VALUE="http://webserver_name/cgi-bin"
```

During form processing, the vtkcgi script places "cgibase"/"http://webserver_name/cgi-bin" into an Input dictionary. When the vtkcgi script processes a template page that contains a &$cgibase, macro reference, the macro is executed and replaced by http://webserver_name/cgi-bin.

Input Data macro

The syntax used for the Input Data macro is

&$xnnn_<label>,

The Input Data macro allows the template page to access data that is passed into the IVR application from the corresponding static Web page. The application developer is able to maintain and display data without repeatedly prompting the Web user each time a Web page is transitioned.

During form processing, the vtkcgi script stores each field, tagged with the name following the xnnn_<label> naming convention, as a NAME/VALUE pair in the Input dictionary. When processing the template page, the vtkcgi script executes any occurrence of the Input Data macro.

If the NAME/VALUE pair exists in the Input dictionary, the macro is replaced by the VALUE. Otherwise, the macro reference is removed and a value is not referenced in the resulting dynamic Web page.

For example, in a static Web page, the following information is contained within a form:

<INPUT TYPE=text NAME="x000_acct" VALUE="1234">

During form processing, the vtkcgi script places "x000_acct"/"1234" into an Input dictionary. When the vtkcgi script processes a template page that contains &$x000_acct, macro reference, the macro is executed and replaced by the value 1234.

Output Data macro

The syntax used for the Output Data macro is:

```
&$ynnn_<label>,
```

The Output Data macro allows the template page to access data that is returned from the IVR application. The application server stores data from the IVR application as NAME/VALUE pairs into an Output dictionary that is returned to the `vtkcgi` script. When processing the template page, the `vtkcgi` script executes any occurrence of the Output Data macro.

If the NAME/VALUE pair exists in the Output dictionary, the macro is replaced by the VALUE. Otherwise, the macro reference is removed, and a value is not referenced in the resulting dynamic Web page.

For example, a template page contains the following information:

```
<p> Your phone number is &$y000_phone, </p>
```

When the `vtkcgi` script processes the template page, the `&$y000_phone,` macro reference is executed and replaced by the corresponding value, `5551114444`, retrieved from the Output dictionary.

Iterator macro

The Iterator macro uses the following syntax:

```
&%name,
```

where *name* is one or more alphanumeric characters.

When processing the template page, the `vtkcgi` script executes any occurrence of an Iterator macro with the value of the current state of the corresponding loop iterator tagged by name that exists in the template page.

If the loop iterator does not exist with a particular name, the macro reference is removed and a value is not referenced in the resulting dynamic Web page. For an example on using the Iterator macro, see “Loop Iterator construct” on page 5-27.

Compound macro

A Compound macro concatenates two or more data fields and uses the following syntax:

`&$`*`name1`*`+[$%]`*`name2`*`+…+[$%]`*`name`*`N,`

where each *`name`* is one or more alphanumeric characters.

During template expansion, the `vtkcgi` script treats the Compound macro as identical to the Simple Data macro except that the tag is formed by the concatenation of the data fields. The data field concatenation rules are explained using the following sample data.

Table 5-1: Sample Input dictionary

Tag	Name	Value
x000_account	000	1234
x001_pin	001	987

Table 5-2: Sample Output dictionary

Tag	Name	Value
y000_name	000	John Smith
y001_phone_num	001	9995551212
y002_zip_code	002	01919

Table 5-3: Sample Iterator values

Iterator	Value
acounter	0
bcounter	2

- If *name*N is formed using either the Input or Output Data format, *$name*N is executed and replaced by the corresponding VALUE from the appropriate dictionary (for example, $x000_pin is executed and replaced with 987 from the dictionary in Table 5-1). Note that the $ is used only with a valid Input or Output data reference.
- If *name*N is formed using a valid Iterator name, %*name*N is executed and replaced by the current value of the Iterator (for example, %bcounter is executed and replaced with 2 using Table 5-3). Note that % is used only with a valid Iterator reference.
- The value *name*1 is formed using either the Input or Output Data format, and results in &$*name*1 expanding to its corresponding NAME representation in the dictionary. This NAME is numerically added to the remaining portion of the Compound macro. The resulting value points to a new NAME in the same dictionary. The new NAME is then replaced with its corresponding VALUE.

  For example, &$y000_name+%bcounter, is executed to 000 + 2 and replaced by 01919, the VALUE for the 002 entry from the dictionary in Table 5-2.

If the Compound macro contains or results in an invalid expansion, the macro reference is removed and a value is not referenced in the resulting dynamic Web page.

Image Button macro

The syntax used for the Image Button macro is:

x000@nnn_<label>,

where

- nnn is a numerical ordering of each Image Button macro
- <label> can be any additional label you may want to provide. This label is not required but will improve the readability of your Web page and therefore is highly recommended.

Rather than relying on the Web user to input data into a field such as a text box, an image can be used that represents data to be passed to the IVR application. The Image Button macro is contained with a form. The data that the Image Button represents is passed with the form to the vtkcgi script when the Web user clicks on the corresponding image.

For example, the following information is contained in a form of a Web page:

```
<INPUT TYPE=image NAME="x000@1_menu_balance"
SRC="bal.gif">

<INPUT TYPE=image NAME="x000@2_menu_xfer" SRC="xfer.gif">
```

Note that the ordering is important. The Web page is presented to the Web user, and the Web user clicks on the image represented by "`xfer.gif`". The form is submitted to `vtkcgi` script and the NAME/VALUE pair, `000/1`, is stored in the Input dictionary. The first MENU cell executed in the IVR application specified by the Web page is passed the VALUE of 1.

Template Page control constructs

This section describes two control constructs available to the application developer for use within the template page. These constructs provide control mechanisms for accessing data that use the methods described in the section that discusses Template Page data fields.

Loop Iterator construct

The Loop Iterator construct is useful for processing large amounts of sequential data that need to be presented to the Web browser which is an ideal way for populating an HTML table. The Loop Iterator construct uses one of the following two formats:

```
<!--%foreach x $name>
    body
</%foreach-->
```

OR

```
<!--%foreach x val1 val2 val3>
    body
</%foreach-->
```

where

- `x` is a string composed of one or more alphanumeric characters
- `name` is a NAME from either the Input or Output dictionaries
- `val1,2` or `3` is a string composed of one or more alphanumeric characters
- *`body`* can consist of Simple Data, Input Data, Output Data, or Iterator Data as singular macros or as a Compound macro

When the template page containing a Loop Iterator construct is processed, the `vtkcgi` script executes the contents of the body for each iteration of the loop controlled by the value of `x`. Table 5-4 contains information that is used to illustrate how the Loop Iterator construct is used.

Table 5-4: Example Output dictionary

Tag	Name	Value
y000_offsets	000	0 2 4
y001_name_1	001	John Smith
y002_acct_1	002	1234
y003_name_2	003	Jane Monty
y004_acct_2	004	8765
y005_name_3	005	Bob Hunt
y006_acct_3	006	5656

The following is an example of a Loop Iterator construct based on the data contained in Table 5-4.

```
<!--foreach offset $y000_offsets>
  <tr><td>&%offset,</td>
  <td>&$y001_name+%offset,</td>
  <td>&$y002_acct+%offset,</td></tr>
</foreach->
```

Using the rules described in the "Template Page data fields" section and the data provided in Table 5-4, the `vtkcgi` script executes the construct. Template expansion results in the following lines appearing in the dynamic Web page:

0	John Smith	1234
2	Jane Monty	8765
4	Bob Hunt	5656

Conditional construct

The Conditional construct provides the template page with the ability to perform different tasks based on whether a tag exists in its corresponding data dictionary. The Conditional construct uses one of the following two formats:

```
<!--%ifdef name> body1 [</%else> body2] </%if-->
<!--%ifndef name> body1 [</%else> body2>] </%if-->
```

where

- *name* represents a NAME/VALUE pair found in either the Input or Output dictionary
- `body` represents one or more lines of HTML or macros to be executed

During template expansion, the vtkcgi script determines if the *name* exists in the appropriate data dictionary. If the *name* is found, `%ifdef` *name* evaluates to TRUE and `body1` is executed. Also, if the name exists, `%ifndef` *name* evaluates to FALSE and `body2` is executed. For example, the following Conditional construct appears in a template page:

```
<!--%ifdef y000_phone_number >
   <p>Your phone number was located!</p>
</%else>
   <p>Unable to locate phone number in our information
   database!</p>
</%if-- >
```

If the `vtkcgi` script locates a `000` NAME entry in the Output dictionary, the message "`Your phone number was located!`" is displayed to the Web user through the dynamic Web page. If the NAME is not found, the message "`Unable to locate phone number in our information database!`" is displayed to the Web user.

Chapter 6: Diagnostics

Overview

The NTTEST software is installed during the system installation, and is resident on the system. It includes diagnostics for testing various applications on your system, including your ASR configuration. For further information about NTTEST, refer to Chapter 24, "Verification test (NTTEST)", in the *Installation, Upgrade, and Maintenance Guide*.

The current location of the NTTEST Web page is by default on the AP (MRS/AP) in `/u/nortel/www/htdocs`. All three NTTEST Web files have to be placed on the Web server by the Web master. The Web master must supply the URL of `index.html`, which is the first page of the Web test. The other two pages (`default.html` and `results.html`) are "reply" pages. They all have to reside at the same "URL - level" or in the same directory, unless you modify the `NTTEST_WEB.vpf` file.

> ***Note:*** See "Determining the target display" on page 5-10 for a procedure to point the Perl scripts to your Web server and `cgi-bin` directory.

Testing your Web setup

It is assumed that those testing the ASR applications are familiar with standard system administration procedures, the details of which are documented in the *System Administration Guide*.

1 Load/run NTTEST and its applications

Run the script `nttest_str`. This loads NTTEST and all its sub applications.

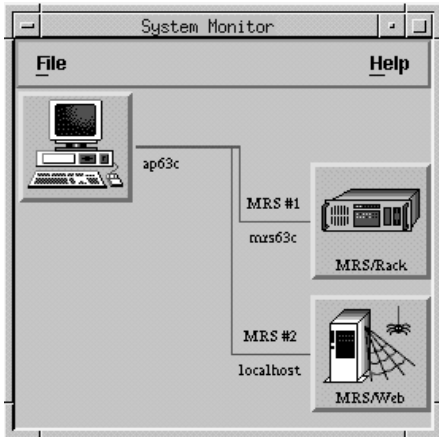
2 Verify the channel alignment

Your system must have been configured already. It should be set to Ring trip, Input, and LoopStart Initialization type as the preferred configuration. Any telephone company changes required would be done through editing of the trunk type files.

a Access the System Administration pull-down menu by pointing and clicking mouse button 1 on the System Administration icon (MRS box with arrows) on the control panel.

b Select the System monitor menu option which will open the System Monitor administration menu. When the outline of a box shows up, move the box to a space on the desktop where you can view it. Select mouse button 1 to show the system monitor.

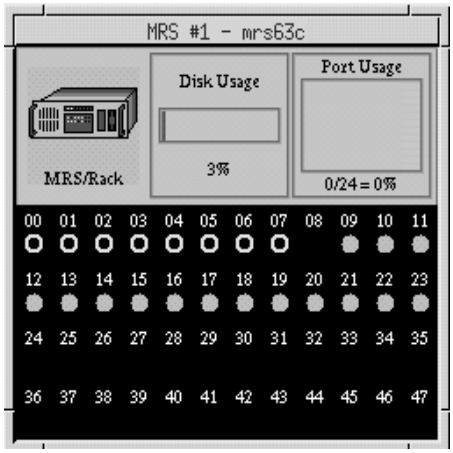
Figure 6-1: System monitor



Note: For more information on how the System Monitor works, please refer to the Release 4.0 *Symposium OPEN IVR System Administration Guide*.

c Select the MRS icon to display the system status window (Figure 6-2). This window contains information on the various channels status.

Figure 6-2: Status Information



Gray Ring means outgoing. Gray circle means incoming on the channel monitor. When the application is loaded, channels assigned, and the application started, the gray circle will change to a yellow ring.

3 Test the Web setup

a Load your Web browser of choice (you can use any Web browser (that supports the protocols defined) on any machine anywhere on the network.

b Connect to your Web server (the location of the URL is supplied by your Web master).

c Enter into the browser any pair of numbers and click on the submit button.

The Web page changes, is replaced with the results page, and shows the calculated sum. (If your system has the "debug flag" turned on, you get additional information displayed with the result.)

4 Stop the test application

To stop the test application, run the `NTTEST_stp` script. This will unload NTTEST and all the sub applications.

5 Exit Application Management

a To exit Application Management, press <F4> to access the Application Management main menu.

b Select `Exit`. The System Application management window closes.

Check the results

There should be an entry in the `event.log` file indicating success. All completions of tests are recorded in the directory `/u/nortel/log.d`.

OPEN IVR CGI script diagnostics

The OPEN IVR CGI script can be placed into a debug mode to allow visibility into the Input and Output dictionaries that are generated between the static Web page and the dynamic Web page returned. To enable debug mode, edit the configuration file, `config.pl`, used by the OPEN IVR CGI script. The `config.pl` file is located on your Web server residing in the directory that contains your CGI executables. Use the following procedure to edit the file.

Procedure 6-1: How to edit the configuration file

1 Log on to your Web server as **`root`**.

2 Change directories (cd) to the location where the OPEN IVR CGI script is installed.

3 Open the `config.pl` file with a text editor.

4 Locate the line that begins with `$debug`.

5 Set `$debug` to `0` (zero) to disable the debug mode. Set `$debug` to `1` (one) to enable debug mode.

6 Save your changes.

7 Exit from the text editor.

Each time the OPEN IVR CGI script executes, the script re-reads the configuration file. You do not need to restart the Web server.

When debug mode is enabled, each dynamic page displayed back to the Web user additionally displays the contents of the current Input dictionary, the current Output dictionary, and Web server environment variables.

Appendix A: Accessing the Service Console Interface

You can access the Service Console Interface (SCI) from a dumb terminal, a modem via the RS-232-C, or the application processor via the Ethernet.

See the following sections:

- "Using a terminal to access the SCI" on page A-2
- "Using a modem to access the SCI" on page A-3
- "Using the LAN to access the SCI" on page A-4
- "Accessing the SCI from an MRS/AP" on page A-5

Default account passwords

The default account passwords shipped with the system are as follows:

Account	Password	Comment
admin	admin1	site administrators on MRS, AP, and MRS./AP
service	service1	service account on MRS
user	user1	routine maintenance on MRS

Using a terminal to access the SCI

To use a terminal to access the SCI, follow these steps.

1 Connect a video terminal to the MRS (MRS/AP) system port B using an appropriate communications cable.

2 Set the terminal parameters as follows:

```
baud rate       2400 or 9600
duplex          Full
data bits       8
stop bits       1
parity          none
add LF          off
```

3 Press <Enter>.

The following message is displayed:

```
<system name>
MRS Service Access PortRevision 4.00
Copyright 1991-1997, Nortel Corp. All Rights
Reserved
Login:
```

4 At the login: prompt, enter the appropriate account name. The passwords for these accounts are listed in “Default account passwords” on page A-1.

5 After you log in, the system prompts you for your terminal type. If the default terminal type displayed is correct, press the <Enter> key; otherwise, type in the correct terminal type. (VT100 and ANSI are sample terminal types that may be entered at the TERM> prompt.)

Once this is completed, the SCI main menu is displayed.

Using a modem to access the SCI

Special modem instructions

To allow remote access for support and maintenance, a modem must be attached to your Nortel system. To take advantage of this feature, you must attach the modem between your Nortel system and the telephone network.

Nortel has tested and found the U.S. Robotics Sportster 9600/28.8/33.6 modems compatible.

U.S. Robotics 14.4 modems were not compatible and therefore are not included in this list.

Domestic customers

Domestic (U.S.) customers receive a modem with the Nortel system.

International customers

Because of the different telephone network regulations throughout the world, international (non-U.S.) customers do not receive a modem with the Nortel system. Nortel recommends that international customers consult their local telephone service provider to determine an appropriate modem (U.S. Robotics Sportster 9600/28.8/33.6 or the equivalent) for their installation.

Accessing the SCI by modem

To use the modem to access the SCI, follow these steps:

1 Connect the modem port on your MRS (MRS/AP) system using a standard modem cable.

2 From the remote location, dial in to the modem and wait for an answer. Once the modem is connected, press the <Enter> key.

The following message should display:

```
<system name>
MRS Service Access PortRevision 4.00
Copyright 1991-1997, Nortel Corp. All Rights
Reserved
login:
```

3 At the `login:` prompt, enter the appropriate account name. The passwords for these accounts are as listed in “Default account passwords” on page A-1.

4 After you log in, the system prompts you for your terminal type. If the default terminal type displayed is correct, press <Enter>, otherwise, type in the correct terminal type. (VT100 and ANSI are sample terminal types that may be entered at the TERM> prompt.)

Once this is completed, the SCI main menu is displayed.

Note: On an MRS/AP, if you log in as `root`, a # prompt is displayed. At the # prompt, type `/usr/vrs/tools/sci -b /usr/vrs` and press <Enter> to access the SCI menu.

Using the LAN to access the SCI

To access an MRS across a LAN, the MRS (MRS/AP) must be connected to either an Ethernet or Token Ring network, and the host that will access the MRS must support TCP/IP and telnet (or rlogin).

To access the SCI using the Ethernet, follow these steps:

1 At the host command line, enter **`telnet 128.1.1.111.`**

Note: If the Internet address for your MRS (MRS/AP) has been changed, enter the correct Internet address instead of the default address shown. This address is noted on your equipment list.

If the host is an X Window-based UNIX workstation, Nortel recommends that this command be issued from an xterm window.

Once the session is connected, the following message should display:

```
login:
```

2 At the login prompt, enter the appropriate account name. The passwords for these accounts are listed in “Default account passwords” on page A-1.

3 After you log in, the system prompts you for your terminal type. If the default terminal type displayed is correct, press <Enter>; otherwise, type in the correct terminal type. (VT100 and ANSI are sample terminal types that may be entered at the TERM> prompt.)

Once this is completed, the SCI main menu is displayed, followed by the `/sci>` prompt.

Note: On an MRS/AP, if you log in as `root,` a # prompt is displayed. At the # prompt, type **`/usr/vrs/tools/sci -b /usr/vrs`** and press <Enter> to access the SCI menu.

Accessing the SCI from an MRS/AP

The modem method (page A-3) and the telnet method (page A-4) can be used to access the SCI from an MRS/AP.

Also, if you have direct access to the console of the MRS/AP system, you can log in and access the SCI by following these steps.

Press the <ALT>, <CNTRL>, and <F1> keys to access the command-line screen.

1 At the `login:` prompt, enter the appropriate account name. The passwords for these accounts are listed in "Default account passwords" on page A-1.

2 After you log in, the system prompts you for your terminal type. If the default terminal type displayed is correct, press <Enter>; otherwise, type in the correct terminal type. (VT100 and ANSI are sample terminal types that may be entered at the TERM> prompt.)

Note 1: If you log in as `root`, a # prompt is displayed. At the # prompt, type **`sci.start`** and press <Enter> to access the SCI menu.

Glossary

This glossary provides definitions of many of the terms as used in this guide. Some of these terms may have additional meanings not described here.

μ-law

Aso called mu-law. A form of Pulse Code Modulation (PCM), alternative to A-law. μ-law is used for the B channel of T1 streams, and is the commonly used North American standard.

A-law

A form of Pulse Code Modulation (PCM), in which the code word maps nonlinearly to the signal value. It is the standard compression algorithm used in digital communications systems of the European digital hierarchy. A-law code words are 8 bits wide. A-law is used for the B channel in ISDN.

acoustic coupler

A device (such as the Audio Interface Unit) used to interface between sound equipment and a port on the MRS/AP.

Administration Tools

A set of tools, accessible from the Symposium OPEN IVR Control Panel, used to configure and control the Symposium OPEN IVR system.

ADSI

Analog Display Services Interface. The Bellcore standard defining the protocol used for the flow of information between a piece of telephone equipment (such as a switch, server, or voice mail) and a subscriber's telephone, PC, data terminal, or other device with a screen.

agent

A person whose primary role is to handle customer calls that are transferred from a queue within a call center.

Analog Display Services Interface

See ADSI.

ANI

Automatic Number Identification. A protocol implemented in some trunk cards that records the calling number.

AP

Application Processor. A computer or workstation running the AP software in a distributed configuration. Symposium OPEN IVR combines one or more Multimedia Resource Servers (MRSs) with a general-purpose Application Processor (AP) to provide a versatile application development and run-time environment. The AP provides the environment for developing, managing, and controlling the IVR applications.

AP software

Nortel's premiere voice application development and management software package that runs on the Application Processor (AP). Application development capabilities are a licensed feature of AP software, through IVR Developer software.

Application Programming Interface. Software libraries that provide a standard access to services provided by a software package.

application

1 In Symposium OPEN IVR, a customized program that controls activity on one or more telephone trunks connected to an MRS voice system.

2 On a host computer, any type of program that carries out a task.

application developer

A person who creates Symposium OPEN IVR applications.

Application Editor

A utility used to graphically create or edit interactive voice response applications.

application processor

See AP.

applicationid

A required hidden field defined in the static Web page that points to the OPEN IVR application to be executed. The name of the IVR application matches the name loaded by the Application Manager on the application server. Do not include the `.vpf` extension.

Automatic Number Identification

See ANI.

branch

A pathway from one cell in an application to the next cell.

buffers

A part of computer memory used to store information from one cell and pass it to another cell.

call flow

A diagram of a Symposium OPEN IVR application.

caller

A person whose phone call is received or originated by a Symposium OPEN IVR application.

cell

The basic element of an application. Each cell causes some action to be performed—for example, playing a prompt to a caller. Each cell has a set of branches to other cells. After the cell performs its action, it determines which branch the application should follow to the next cell.

channel

1 A telephone trunk within a cluster of MRSs.

2 A path of communication between the AP and a communications or transmission facility. Channel sequences can span multiple MRSs, and their numbering starts at zero (0). Also referred to as a **port**.

CLI

Call Logic Interpreter. Acts as a virtual machine which manages memory for IVR applications.

CO

Central Office.

COMA cell

A Symposium OPEN IVR cell that cancels a transaction in a VT100 terminal session or with an SNA host. It does not terminate the session itself.

COMI cell

A Symposium OPEN IVR cell that sends input to an SNA host application or a VT100 terminal session using the TRS process.

COMO cell

A Symposium OPEN IVR cell that receives output from an SNA host application or VT100 terminal session through the TRS process.

compound macro

A macro provided by the OPEN IVR CGI script that allows a template page to construct Simple Data fields through concatenation.

conditional construct

A control construct provided by the OPEN IVR CGI script that allows a template page to conditionally control segments of the dynamic Web page generations based on whether data is found in the Output dictionary.

config.pl

The configuration file used by the OPEN IVR CGI script that contains the base path locations of the CGI programs and the HTML documents. The file is placed onto the Web server during the AP software Web Option installation.

Control Panel

The Symposium OPEN IVR panel containing the buttons from which you can access the Administration, Information Database Editor, Report, Application, Prompt Management, and SmartHelp tools.

database

A collection of related information arranged for ease and speed of retrieval.

dftlnextpage

A required hidden field defined in the static Web page that points to the name of the default template page. The default template page is used in cases where the IVR application concludes without pushing a template page through a PDAT cell.

Dialed Number Identification Service

See DNIS.

DID

Direct inward dialing. A protocol used by some telephone equipment that allows callers to dial directly to an extension within a telephone system without going through a switchboard.

Direct inward dialing

See DID.

DNIS

Dialed Number Identification Service. A feature of 800 and 900 numbers, in which the number dialed or a portion of it is sent to the MRS and stored in the DIGITS buffer in OPEN IVR.

DTMF

Dual Tone Multi-Frequency. Touchtone (push button) dialing. DTMF is used to encode digits over analog telephone lines. Applications can collect information from callers by having them press telephone keys to create DTMF tones.

Dual Tone Multi-Frequency

See DTMF.

EXEC

The cell that executes an OPEN IVR application on the current channel.

field

The specific location of information (data) within a record in a relational database.

GDAT

Play Prompts and Get Data. A cell that allows you to play one or more prompts and collect up to 16 digits of input from the caller.

host computer

A networked computer that provides applications and services to other networked computers. The VT100 Gateway product sends information to and receives information from the host computer via the TRS.

Host link

The ability of Symposium OPEN IVR to communicate with other host computers supported by one of the OPEN IVR gateway products. *See also* COMA cell, COMI cell, and COMO cell.

Image Button macro

A macro provided by the OPEN IVR CGI script that allows a Web page to use images as a method for providing data for the IVR application.

Information Database

See database.

Information Database Editor

A utility used to create a message or information database, and to edit an information database.

Input Data macro

A macro provided by the OPEN IVR CGI script that allows a template page access to the data contained in the current Input dictionary.

Input dictionary

A container for data being passed from the Web browser to the IVR application by the OPEN IVR CGI script.

IRQ

Interrupt Requests (also called hardware interrupts). They are a way that a device signals the data bus and the CPU that it needs attention.

Iterator macro

A macro provided by the OPEN IVR CGI script that allows a template page to access the value of an iterator during expansion.

IVR

Interactive Voice Response.

IVR Developer

A licensed component of AP software that provides application development tools to graphically create or edit interactive voice response applications.

license file

A file which reflects the various product licenses purchased for your system. Whenever you modify any of the options available on an MRS with Symposium OPEN IVR MRS software 4.0 or greater, you need to load a new license file.

Loop Iterator construct

A construct provided by the OPEN IVR CGI script that gives a template page the ability to process large amounts of Output dictionary data without needing to reference each individually.

loopstart

A method of signalling on the trunk in which a supervisory signal is given (by taking the phone off the hook). Loopstart seizes the line by bridging the resistance of the tip and ring wires of the telephone line to obtain a dial tone. Loopstart signalling is normally used for single lines or key systems.

MF tones

Multifrequency tones. These are tones used by ANI. Symposium OPEN IVR applications can receive and store MF digits when the ANI option is in use.

MRS

Multimedia Resource Server. Symposium OPEN IVR combines one or more Multimedia Resource Servers (MRSs) with a general-purpose Application Processor (AP) to provide a versatile application development and/or run-time environment. The MRS provides the telephony, information processing (voice, fax, speech recognition, and text-to-speech) environment.

MRS/AP

A single unit that runs a “folded” configuration; that is, a configuration that includes both the Multimedia Resource Server and the Application Processor. The MRS/AP contains the functionality of the other platforms, but in a single hardware unit.

MRS/Rack

A unit comprising a rack-mountable Application Processor (AP) and a rack-mountable Multimedia Resource Server (MRS). Multiple MRS/Racks can be installed in cabinets and stacked to take up less room.

MRS/Tower

The MRS/Tower platform is comprised of an Application Processor and a Tower Multimedia Resource Server.

multifrequency tones

See MF tones.

Multimedia Resource Server

See MRS.

mu-law

Also called μ-law. A form of Pulse Code Modulation (PCM), alternative to A-law. Mu-law is used for the B channel of T1 streams, and is the commonly used North American standard.

node

A grouping composed of an AP (running Symposium OPEN IVR) connected to one or more MRSs.

Output Data macro

A macro provided by the OPEN IVR CGI script that allows a template page access to the data contained in the current Output dictionary.

Output dictionary

A container for data being passed from the IVR application to the OPEN IVR CGI script.

overlay

A fax overlay allows the superimposing of a TIFF file onto the ASCII fax data file. In Release 4.0 fax overlays are only supported by the GammaLink fax card.

parameter

A variable used to define the function of Symposium OPEN IVR cells. You set parameters in the parameter window for each cell (when using the Application Editor), or in the cell window (when using the character-based Application Editor).

PBX

Private branch exchange. A private phone system allowing communication within a business and between the business and the public telephone system.

PCM

Pulse Code Modulation. A method that uses a sequence of discrete integers as code words to describe a time-varying signal. Each integer code word describes the signal level at a point in time.

PDAT

Play Prompts with Data. A cell that allows you to play prompts to announce the contents of a buffer. The buffer content is read according to the specified type.

PEB

PCM expansion bus. A digital voice bus for sending voice across different voice processing cards and components in the same PC. It is an open platform.

port

A path of communication between the AP and a communications or transmission facility. Port sequences can span multiple MRSs, and their numbering starts at zero (0). Also referred to as a **channel**. *See also* Trunk.

private branch exchange

See PBX.

prompt

A voice recording that helps lead a caller through a Symposium OPEN IVR application.

Prompt Management tools

A set of tools, accessed from the Symposium OPEN IVR Control Panel, that is used to record, load, and update prompts.

SCbus

Signal Computing bus—the standard bus for communication within an SCSA node. It is a serial time division multiplexed bus for carrying information between hardware devices in a signal processing mode. SCbus features a hybrid bus architecture consisting of a serial message bus for control and signalling, and a 16-wire TDM data bus.

session

1 A connection to a host computer as defined in the `trs.conf` and `vt100.ctl` files, representing a terminal connection.

2 A connection between two systems for the exchange of user data. Sessions are between logical units (LUs).

Simple Data macro

A macro provided by the OPEN IVR CGI script that allows a template page to access the value of the Input dictionary not associated with data designated for the IVR application such as the required field, applicationid.

socket

An end point (portal) from one TCP/IP application to another.

striping

A feature which allows the addition of images such as logos and signatures with ASCII text on a fax cover page. Embedded text commands in the Fax Data file designate the placement of images and text and control some of the aspects of text layout as well.

subscriber

A person serviced by a voice processing application (such as voice mail).

system administrator

A person responsible for configuring MRSs, installing and running Symposium OPEN IVR applications, managing prompts, and running reports.

Telco

Telephone company.

1 An ASCII file used by the TRS process to manage the SNA terminal session.

2 An ASCII file used by the TRS process to manage the VT100 terminal session.

3 A binary pattern representing a word or utterance.

template expansion

A procedure that is performed by the OPEN IVR CGI script based on a specified template page. Expansion is relative to the current Input and Output dictionaries. Expansion evaluates the contents of the template page to generate a dynamic Web page.

template page

A file containing HTML constructs and OPEN IVR CGI macros that is used during template expansion for generating a dynamic Web page.

Text-to-speech

Text-to-speech (TTS) is the process of converting text—usually in the form of ASCII files—to speech output. It is used in applications where it is inconvenient or impossible to record a human speaker.

TIFF

Tagged Image File Format. TIFF provides a way of storing and exchanging digital image data. Originally developed by Microsoft Corp., Aldus Corp., and major scanner vendors to help link scanned images with popular desktop publishing applications, it is now used in many different of software applications, ranging from medical imagery to fax modem data transfers. (It does not mean “Took It From a Fotograph”.)

Trunk

A communication line between two switching systems, frequently referred to as a port. Trunk numbering starts at 1.

vtkcgi.cgi

OPEN IVR Common Gateway Interface (CGI). This CGI script provides the communication between the Web browser and the IVR application.

Web channel

A channel configured on the Web MRS for World Wide Web access into the IVR application through a “call” mechanism.

wink

A momentary interruption in single frequency tone that indicates the distant central office (CO) is ready to receive the digits that were just dialed.

XAE

X-Window Application Editor. Another term for the Application Editor.

Index

A

accessing SCI
from a LAN A-4
from a modem A-3–A-4
from a terminal A-2
from an MRS/AP A-5
ACTION 5-7
advantages of Web option design 1-1
AP requirements 2-1
AP software
applications
and hidden fields 1-5
process for developing 5-2
licensing 2-5
media contents 2-6
preinstalled 2-5
release media 2-5
Web applications
error causing cells 4-2
application developers, skills required 5-1
applicationid field
discussed 5-6
appropriate modems for system access A-3

C

call flow
graphical example 5-13
call into IVR 1-4
call types, and ISWEB 5-12
capabilities
available to IVR and Web applications 5-20
cells
and ISWEB 4-2
listed for IVR and Web 4-3
telephony
events while IVR running 1-4
which abort Web applications 4-2
CGI program
functions 5-20
compound macro
syntax 5-25
concatenation rules, for data fields 5-26
config.pl file 6-4
configuring
Web MRS 3-2–3-3
Web sockets 3-4–3-7
confirming
Web MRS configuration 3-7–3-8
Web sockets 3-7–3-8
control constructs 5-29
conditional 5-29
loop iterator 5-27–5-28
example 5-28
on template page 5-27–5-29
convention for naming input data 5-6
criteria
for IVR example 5-3
customize
error files after installation 2-9

D

data
dictionary 4-1
defined 5-21
field concatenation rules 5-26
more complex in Web calls 5-19
passing to IVR application 5-12
debug mode 6-4
default password settings A-1
default value
of ISWEB 3-9, 5-12
definition
of data dictionary 5-21
of template page 5-14
of Web MRS 3-1
definition of
template 11
design advantages, of Web option 1-1
determining call types with ISWEB 5-12
dfltnextpage field
discussed 5-6
discussion
of applicationid field 5-6
of dfltnextpage field 5-6
of forms 5-5
of Input dictionary 5-21
of INPUT fields 5-5
of ISWEB 3-9, 5-12
of Output dictionary 5-22
dynamic Web page 1-5

E

editing the config file 6-4
error causing cells 4-2
error cells
and ISWEB 4-2
error files
customize after installation 2-9
example
of call flow selection 5-13
of ISWEB buffer 4-4
of Web/IVR interactions 1-2–1-3

F

factors
for AP system load 2-2
for Web server load 2-2
file
config.pl 6-4
files to customize after Web installation 2-9
forms
discussed 5-5

G

GDAT cell
accepts Web input data 5-12
accepts Web inputs 4-1
and ordering 5-12
Web parameters supported 4-1
generalized procedure
Web page design 5-7–5-8
Web template design 5-16–5-17
graphical representation
of IVR example 5-4

H

HANG cell
in Web applications 4-2
hidden fields
use with AP software applications 1-5

I

image button macro
syntax 5-26
input data
macro
syntax 5-15
naming convention 5-6, 5-21
syntax for IVR 5-6
input data macro
syntax 5-23
Input dictionary 1-4, 1-6, 5-5, 5-12, 5-15, 5-21, 6-4
discussed 5-21

NAME/VALUE fields 5-23
sample 5-25
INPUT field, discussed 5-5
installation
customize afterwards 2-9
of Web option 2-7–2-9
installing IMTESTa 6-1
integrating Web and IVR applications 5-2
interaction
between Web and IVR 1-1
as seen from OPEN IVR 1-6
as seen from Web server 1-5
graphically illustrated 1-3
between Web and PSTN 1-1
ISWEB
determining cell usage 4-2
determining type of call 5-12
discussion of 3-9, 5-12
graphical sample 4-4
iterator construct loop
sample values 5-25
iterator construct, loop 5-27–5-28
example 5-28
iterator macro
syntax 5-24
IVR
additional services provided by system 5-2
application example
criteria 5-3
graphical representation 5-4
overview 5-3
applications
suggestions for effective design 5-19
call, how it's established 1-4
capabilities
available to applications 5-20
cells, listed 4-3
example
of Web interactions 1-2–1-3
IVR application
input data syntax 5-6
pushing a template page 5-13
IVR node, as a firewall 1-1
IVR/Web integration
mechanisms needed for success 5-11
IVR/Web interaction
as seen from OPEN IVR 1-6
as seen from Web server 1-5
graphically illustrated 1-3
integrating the applications 5-2
sample scenario 1-4–1-5

L

LAN to SCI access A-4
list of IVR and Web cells 4-3
load
on AP system, factors 2-2
on Web server, factors 2-2
loop iterator construct 5-27–5-28
example 5-28

M

macros
compound 5-25
image button 5-26
input data 5-23
iterator 5-24
output data 5-24
simple data 5-22
mechanisms for IVR/Web integration 5-11
media contents of AP software 2-6
MENU cell
accepts Web inputs 4-1
Web parameters supported 4-1
mode
debug 6-4
modem
appropriate for system access A-3
to SCI access A-3–A-4
MRS requirements 2-1
MRS/AP to SCI access A-5

N

NAME/VALUE fields 5-23–5-24
naming convention for input data 5-6

O

OPEN IVR CGI script 1-4, 1-5, 1-6, 5-7, 5-12, 5-15, 6-4
ordering, and GDAT cell 5-12
output data macro
  syntax 5-15–5-16, 5-24
Output dictionary 1-4, 1-5, 6-4
  discussed 5-22
  sample 5-25
overview
  of IVR example 5-3
  of Web page design 5-5

P

passing data
  rules of operation 5-12
  to IVR application 5-12
passwords
  default settings A-1
PDAT cell 1-4
  Web parameters ignored 4-2
prerequisites
  for Web server installation 2-6
procedure
  for configuring a Web MRS 3-2–3-3
  for configuring Web sockets 3-4–3-7
  for editing config file 6-4
  for generalized Web page 5-7–5-8
  for generalized Web template 5-16–5-17
process
  for developing a Web application 5-2
prompts
  and Web applications 4-1
PSTN 1-1

R

release media for AP software 2-5
requirements
  for AP 2-1
  for MRS 2-1
  for system loads 2-2
  for Web browser 2-2
  for Web server 2-2
Robotics Sportster modem A-3
rules
  for passing data 5-12

S

sample
  Input dictionary 5-25
  Iterator values 5-25
  Output dictionary 5-25
  scenario of Web/IVR interaction 1-4–1-5
sample input
  for Web page design 5-9, 5-17
SCI, accessing from
  a LAN A-4
  a modem A-3–A-4
  a terminal A-2
  an MRS/AP A-5
SCM interface
  and configuring Web sockets 3-4–3-7
services additionally provided by IVR system 5-2
settings
  for default passwords A-1
simple data macro
  syntax 5-22
skills required
  for application developers 5-1
  for Web masters 5-1
socket connection
  to the AP 1-5
software must be licensed 2-5
submitting the form 5-7
suggestions
  for effective IVR applications 5-19
  for effective Web applications 5-18–5-19

syntax
of compound macro 5-25
of image button macro 5-26
of input data macro 5-15, 5-23
of iterator macro 5-24
of IVR input data 5-6
of output data macro 5-15–5-16, 5-24
of simple data macro 5-22
system requirements, listed 2-2

T

telephony
cells
events while IVR running 1-4
extends Web experience 1-2
services
provided by IVR system 5-2
template page 1-6
control constructs 5-27–5-29
data fields
compound macro 5-25–5-26
image button macro 5-26–5-27
input data macro 5-23
iterator macro 5-24
output data macro 5-24
simple data macro 5-22–5-23
defined 5-14
pushed from IVR application 5-13
template, defined 11
terminal to SCI access A-2
testing
installing IMTESTa 6-1

V

vtkcgi script 5-5, 5-21, 5-23, 5-24, 5-25, 5-26, 5-27, 5-28, 5-29
functions of 5-21
special macros
compound macro 5-25
image button macro 5-26
input data macro 5-23
iterator data macro 5-24
output data macro 5-24
simple data macro 5-22

W

Web
application
process for developing 5-2
applications
and cell prompts 4-1
and GDATcell parameters 4-1
and HANG cell 4-2
and MENU cell parameters 4-1
and PDAT cell parameters 4-2
error causing cells 4-2
suggestions for effective design 5-18–5-19
browser requirements 2-2
calls
can handle more complex data 5-19
capabilities
available to applications 5-20
cells, listed 4-3
design procedure
for page 5-7–5-8
for template 5-16–5-17
example
of IVR interactions 1-2–1-3
masters, skills required 5-1
MRS
confirming configuration 3-7–3-8
confirming Web sockets 3-7–3-8
defined 3-1
procedure for configuring 3-2–3-3
option
design, advantages of 1-1
installation procedure 2-7–2-9
post installation adjustments 2-9
page
and forms 5-5
design overview 5-5
passes data dictionary 4-1
passing data to IVR application 5-12

sample input 5-9, 5-17
submitting the form 5-7
parameters
ignored by PDAT cell 4-2
supported by GDAT cell 4-1
supported by MENU cell 4-1
server
installation prerequisites 2-6
requirements 2-2
sockets
procedure for configuring 3-4–3-7
template
generalized design procedure 5-16–5-17
users benefit from telephony 1-2
Web/IVR integration
mechanisms needed for success 5-11
Web/IVR interaction 1-1
as seen from OPEN IVR 1-6
as seen from Web server 1-5
graphically illustrated 1-3
integrating the applications 5-2
sample scenario 1-4–1-5
Web/PSTN interaction
advantages of 1-1

NORTEL
NORTHERN TELECOM

Reader Response Form

Symposium OPEN IVR Release 4.0
Web Option Guide
NTVE8210/A0732109

Tell us about yourself:

Name: _____

Company: _____

Address: _____

_____

Occupation: _____ **Phone:** _____

1. What is your level of experience with this product?

   ☐ New User ☐ Intermediate ☐ Experienced ☐ Programmer

2. How do you use this book?

   ☐ Learning ☐ Procedural ☐ Reference ☐ Problem solving

3. Did this book meet your needs?

   ☐ Yes ☐ No

   If you answered No to this question, please answer the following questions.

4. What chapters, sections, or procedures did you find hard to understand?

   _____

   _____

   _____

5. What information (if any) was missing from this book?

   _____

   _____

   _____

6. How could we improve this book?

   _____

   _____

   _____

Please return your comments by fax to (416) 597-7104, or mail your comments to:
Toronto Information Products, Nortel, 522 University Avenue, 14th Floor, Toronto, ON, Canada M5G 1W7.

NORTEL
NORTHERN TELECOM

Reader Response Form

Reader Response Form

Symposium OPEN IVR

Web Option Guide

Toronto Information Products
Nortel
522 University Avenue, 14th Floor
Toronto, Ontario, Canada
M5G 1W7

© 1998 Northern Telecom
All rights reserved

Information is subject to change without notice. Northern Telecom reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

Symposium, OPEN IVR, and Nortel are registered trademarks of Northern Telecom Limited. DEC and VT 420 are registered trademarks of Digital Equipment Corporation. HP, LaserJet, and ThinkJet are registered trademarks of Hewlett-Packard Company. X Window System and X are trademarks of the Massachusetts Institute of Technology. Motif is a trademark of Open Software Foundation Inc. UNIX is a registered trademark of Novell, Inc.

Product release: 4.0
Document release: Standard 1.0
Date: July 1998

Printed in the United States of America

NORTEL
NORTHERN TELECOM